

GABRIEL LEONARDI BERTAZZI

Desenvolvimento de um aplicativo móvel para
monitoramento de vagas e estacionamentos em locais
públicos e privados.



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA

2019

GABRIEL LEONARDI BERTAZZI

Desenvolvimento de um aplicativo móvel para monitoramento de vagas e estacionamentos em locais públicos e privados.

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Mecatrônica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **ENGENHEIRO MECATRÔNICO**.

Orientador: Prof. Dr. Luciano Vieira Lima

UBERLÂNDIA

2019

GABRIEL LEONARDI BERTAZZI

Desenvolvimento de um aplicativo móvel para monitoramento
de vagas e estacionamentos públicos e privados.

Trabalho de Conclusão de Curso **APROVADO**
pelo Colegiado do Curso de Graduação em Engenharia
Mecatrônica da Faculdade de Engenharia Mecânica da
Universidade Federal de Uberlândia.

Banca Examinadora:

Prof. Dr. Luciano Vieira Lima / (Orientador)

Prof. Dra. Vera Lúcia D.S. Franco

Prof. Dr. Keiji Yamanaka

UBERLÂNDIA

2019

“Nós escolhemos ir à lua! Nós escolhemos ir para a lua nesta década e fazer outras coisas, não porque sejam fáceis, mas porque são difíceis. Porque esse objetivo servirá para organizar e medir o melhor de nossas energias e habilidades, porque esse desafio é aquele que estamos dispostos a aceitar, um que não queremos adiar e um que pretendemos vencer.”

(John F. Kennedy, 12 de setembro de 1962)

Agradecimentos

Agradeço a deus pela oportunidade de estar vivo, saudável, e de ter conseguido estudar em uma faculdade federal na área de engenharia, uma das áreas mais disputadas entre os estudantes.

Agradeço aos meus pais e ao meu irmão pelo apoio que tive durante a faculdade, além do amor, carinho e da educação me dada na infância.

Agradeço ao meu professor tutor Luciano Vieira Lima que me ajudou bastante durante todo o meu projeto, sempre sendo bastante compreensivo e me guiando quando necessário.

Agradeço também ao Thales Oliveira Lima, que me ajudou muito com a parte de programação do projeto, me auxiliando a encontrar erros, e também me ajudando com a lógica.

Agradeço a todos os meus amigos criados no âmbito da universidade, amigos que compartilhei muitos momentos felizes e difíceis, o que torna a experiência da faculdade mais bonita e rica.

Por fim Agradecer à Faculdade de engenharia mecânica e a Universidade federal de Uberlândia, que me proporcionaram ensino de qualidade que serão muito importantes para a minha vida.

Bertazzi, G.L, **Desenvolvimento de um aplicativo móvel para monitoramento de vagas e estacionamentos públicos e privados**. 2019. 141f. Trabalho de conclusão de curso (Bacharel) – Curso de graduação em Engenharia Mecatrônica, Universidade Federal de Uberlândia, Ubelândia 2019.

Resumo

O mundo está muito rápido, o tempo tem se tornado uma moeda preciosa hoje em dia, as ruas estão cada vez mais lotadas de carros andando de um lado para o outro levando pessoas para os seus compromissos, e isso tem aumentado cada vez mais o transito e diminuído o número de vagas para que as pessoas estacionem os seus carros. Estes dois problemas têm atrapalhado e irritado muito as pessoas do século 21, mas e se tivesse alguma maneira de ajudar as pessoas solucionar o segundo problema, fazendo com que o próprio celular deste indivíduo a ajudasse a encontrar de maneira fácil um local para estacionar, sem estresse? Este aplicativo de celular é o que será tratado no decorrer deste trabalho.

Palavras Chave: Corona SDK, Desenvolvimento de aplicativo Mobile, Modelo Canvas, Modelo de negócio, Pesquisa de mercado, Matriz GUT,

Bertazzi, G.L, **Development of a mobile application for monitoring public and private parking and parking**. 2019. 141p. Final Course Assignment (Bachelor) – Mechatronics Engineer Graduation, Universidade Federal de Uberlândia, Uberlândia, 2019

Abstract

The world spinning too fast, time has become a precious currency these days, the streets are increasingly crowded with cars moving around bringing people to their appointments, and this has increasingly increased traffic. and decreased the number of spaces for people to park their cars. These two problems have been upsetting and annoying people of the 21st century a lot, but what if it had some way of helping people solve the second problem by having this individual's own cell phone help her easily find a parking spot without stress? This mobile app is what will be dealt with in the course of this work.

Keywords: Corona SDK, Mobile App Development, Canvas Model, Business Model, Market Research, GUT Matrix.

Sumário

1.	Introdução.....	14
1.1.	Objetivo.....	15
1.2.	Justificativa.....	15
1.3.	Condições de Contorno.....	16
1.4.	Estrutura do Trabalho.	16
1.5.	Estrutura do Aplicativo.	17
1.5.1.	Vagas Ociosas em prédios ou moradias.....	18
1.5.2.	Vagas particulares de estacionamentos particulares.	18
1.5.3.	Vagas Públicas.....	19
2.	Fundamentação Teórica.....	20
2.1.	Modelo de negócio Canvas.....	20
2.1.1.	O que?.....	21
2.1.2.	Para Quem?	22
2.1.3.	Como?.....	23
2.1.4.	Quanto Custa?	23
2.2.	Pesquisa de mercado	24
2.3.	Matriz GUT	25
2.4.	Paradigma computacional.....	27
2.4.1.	Programação orientada a objetos (POO):	27
2.4.2.	Programação Imperativa:	28
2.3.4	Programação visual:	28
2.3.5	Programação Genérica:.....	29
2.4	Escolha da Plataforma de programação Mobile.....	29

2.4.1 Java:	29
2.4.2 C++:	30
2.4.3 Corona SDK:.....	30
2.4.4 APP Inventor.....	31
2.3.5 Plataforma Escolhida:	33
2.5 Deep Learning	33
I. Conceito:	33
3. Criação do produto.	35
3.1. Criação do modelo de negócio.....	35
3.1.1 Para quem?	35
3.1.2 O que e para quem?	36
3.1.3 Como?	39
3.1.4 Quanto Custa?.....	43
3.2. Pesquisa de Mercado	45
3.2.1. As Questões, ...	46
3.2.2. Resultados Obtidos.	51
Seção 1:	51
Seção 2:	52
Seção 3:	53
Seção 4:	57
Seção 5:	58
3.3. Definição das atividades e desenvolvimento da matriz GUT.	59
3.3.1. Definição das atividades.....	60
3.3.2. Matriz GUT.	62
4. Desenvolvimento do aplicativo e aplicações futuras.....	65
4.1. Como O App funciona.....	65
4.2. Implementações Futuras.	83

4.2.1.	Como encontrar vagas em locais públicos.....	83
4.2.2.	Problemas que devem ser levados em consideração para a criação do algoritmo em Deep Learning.....	84
5.	Conclusão.....	90
6.	Bibliografia.....	92
7.	Anexos.....	94
7.1.	Programas feitos no Corona (linguagem lua).....	94
7.2.	Programas feitos para o banco de dados Online (linguagem php)	135

ÍNDICE DE ILUSTRAÇÕES

Figura 2.1 - Quadro que representa o modelo canvas	21
Figura 2.2 - Matriz GUT com notas explicadas.	26
Figura.2.3 - Framework do programa Corona SDK	31
Figura.2.4 - Framework do APP inventor	32
Figura 2.5 - Estrutura que representa uma rede neural de um algoritmo de Deep Learning	34
Figura.3.1 - Segmento de clientes preenchido no canvazizer	36
Figura.3.2 - Segmento da proposta de valor	37
Figura.3.3 - Segmento de relacionamento com o cliente.	38
Figura.3.4 - Como foi preenchido o segmento canais	39
Figura.3.5 - Como foi preenchido o segmento de atividades chave.	41
Figura.3.6 - Preenchimento do segmento de recursos chave	42
Figura.3.7 - Preenchimento do segmento de parceiros chave.	43
Figura 3.8 - Preenchimento do segmento de fontes de receita.	44
Figura 3.9 - Preenchimento do segmento de estrutura de custos.	45
Figura 3.10 - Primeira seção do questionário	46
Figura 3.11 - segunda seção do questionário	47
Figura 3.12 - Seção 3 do formulário	48
Figura 3.13 - Seção 4 do formulário	49
Figura 3.14 - Últimas questões das seções 3 e 4	50
Figura 3.15 - seção 5 do formulário	50
Figura 3.16 - Algumas respostas da seção 1 do formulário	51
Figura 3.17 - Resultado de quantas pessoas responderam o formulário tem carros ou não	52
Figura 3.18 - Respostas obtidas na segunda seção do questionário	53
Figura 3.19- Resultados obtidos sobre dificuldade de estacionamento, e local procurado para estacionar	54
Figura 3.20 - Resultados obtidos sobre local de estacionamento, e o quanto é importante estacionar próximo ao local de destino	55
Figura 3.21 - Resultados obtidos das pessoas que preferem ou não utilizar transporte público	56

Figura 3.22 - Resultados obtidos para entender se as pessoas utilizam o gps do celular, e se seriam adeptas a um aplicativo para ajuda-las a encontrar vagas.	57
Figura 3.23 - Resultados obtidos para entender se as pessoas pagariam pelo aplicativo	58
Figura 3.24 - Seção de Receitas com as mudanças vistas necessárias pela pesquisa de mercado.	59
Figura 3.25 - Resultados obtidos na análise GUT	62
Figura 4.1 - Fluxograma de funcionamento do aplicativo	67
Figura 4.2 - Parte principal do aplicativo.	69
Figura 4.3 - Figura que mostra um local marcado no mapa encontrado pela barra de procura	70
Figura 4.4 - Figura que mostra como o usuário pode identificar uma vaga no aplicativo, e as informações da vaga caso o usuário toque nela.	71
Figura 4.5 - Figura que mostra a mensagem de confirmação de reserva de vaga.	72
Figura 4.6 - Alerta de que a reserva foi feita com sucesso.	72
Figura 4.7 - Aba menu aberta antes que o login tenha sido feito.	73
Figura 4.8 - Aba menu aberta após a execução do login	74
Figura 4.9 - Área de cadastro de membros	75
Figura 4.10 - Mensagem avisando que as senhas são diferentes.	75
Figura 4.11 - Mensagem de Erro caso o usuário entre com um CPF inválido	77
Figura 4.12 - Tela de login	78
Figura 4.13 - Mensagem de erro na tela de login	78
Figura 4.14 - Tela de cadastro de cartões	79
Figura 4.15 - Mensagem de erro, caso o CPF seja invalido	80
Figura 4.16 - Tela de registro de endereço	81
Figura 4.17 - Tela aonde o usuário vai selecionar a data da reserva	82
Figura 4.18 - Mensagem de erro caso a pessoa selecione uma data inválida.	82
Figura 4.19 - Figura que mostra bastante espaço livre no qual claramente cabe um carro ou dois carros	84

Figura 4.20 - Espaço livre entre 2 carros, entretanto que não cabe um carro entre eles.	85
Figura 4.21 - Local vago a frente de uma garagem, que não pode ser representado como vaga pelo aplicativo.	86
Figura 4.22 - Placas de lei de transito na ordem: Permitido estacionar e parar, proibido estacionar, proibido estacionar e parar	86
Figura 4.23 - Figura de um carro sendo advertido por desobedecer a placa de trânsito	87
Figura 4.24 - Faixa amarela que diz que é proibido estacionar.	88
Figura 4.25- imagem que mostra linhas tracejadas brancas que permitem que o carro estacione, além de mostrar uma delimitação feita com linhas para cada vaga.	89
Figura 4.26 - Foto de um canteiro central que o aplicativo deve reconhecer.	90

INDICE DE ABREVIACES

GPS: Global Positioning System (Sistema de posicionamento global)

LINA: Laboratrio de inteligncia natural e artificial.

Matriz GUT: Gravidade, Urgncia e Tendncia

POO: Programaco orientada a objetos.

iOS: iPhone Operational System

MIT: Massachusetts institute of technology,

APP: Application (utilizado para falar sobre aplicativos de celular ou computador)

API: Application Programming interface (interface de aplicaco programvel).

1. Introdução

Com a evolução da tecnologia e o desenvolvimento de peças eletrônicas cada vez menores e mais eficientes, o uso de computadores de bolso, também conhecidos como smartphones, tem se tornado algo extremamente comum na sociedade, o que vem causando até alguns males devido ao seu uso abusivo por muitos cidadãos ao redor do mundo. Estes aparelhos eletrônicos trouxeram para a sociedade, uma maneira muito fácil de estar conectado à internet, e através desta conexão, foram desenvolvidos softwares com funções diversas, que se utilizam dos sensores dos smartphones, para facilitar a vida das pessoas; estes softwares ganharam o nome de aplicativos, e podem ser utilizados para fazer a comunicação entre as pessoas por mensagens de texto (WhatsApp®), para pedir um delivery de comida (Ifood®), e até mesmo para fazer o gerenciamento dos trabalhos dos funcionários de uma empresa (Trello®).

Além disso com o desenvolvimento dos hardwares dos smartphones, o processamento e a precisão dos seus sensores, tem se tornado algo inimaginável para quem já trabalhava com computadores a 15 anos atrás. Para dar um exemplo desta precisão, os gps utilizados pelos smartphones atuais, conseguem ter uma precisão de 30cm, o que é muito pouco visto que, o gps tem de monitorar cidades inteiras, e são criados por monitoramento de satélites, ou seja, máquinas fora da terra conseguem monitorar a sua localização com um erro de 30 cm, o que é incrível, e fez com que muitos aplicativos baseados em gps, surgissem, como o Waze® e o Google maps®, dois GPS clássicos gratuitos, o Uber um sistema criado para chamar um motorista particular, que te leva até o seu destino cobrando um valor justo pela distância percorrida, e tantos outros aplicativos que usam deste sistema de gps presente no celular.

Com o mundo atual cada vez mais mobile, e cada vez mais rápido, pequenos problemas como por exemplo fazer, dificuldade para encontrar uma vaga, para deixar o seu carro, próximo aos locais aonde determinada pessoa precisa ir, podem se tornar um grande problema. Por conta disso, e devido aos grandes avanços tecnológicos obtidos na área de aplicativos mobile, será desenvolvido um aplicativo, que tem a função de ajudar

a encontrar vagas para os usuários que utilizaram este aplicativo, com o objetivo de fazer com que este aplicativo se torne um produto competitivo de mercado.

1.1. Objetivo

- Este trabalho tem como objetivo, aplicar algumas das ferramentas da administração para criar um plano de negócios de uma empresa
- fazer uma pesquisa de mercado para verificar se o aplicativo de encontrar vagas seria bem aceito pelo público, e se é possível que ele seja lançado no mercado.
- Independentemente do resultado obtido pela análise de mercado, será desenvolvido um aplicativo protótipo que deve facilitar que pessoas disponibilizem vagas, e estas vagas sejam visualizadas pelos outros usuários do aplicativo e possam ser reservadas por eles.
- Por fim será citada uma maneira de como o aplicativo pode ser melhorado para encontrar vagas em locais públicos, e o que deve ser pensado no momento de criar o algoritmo para isso.

1.2. Justificativa.

Com o aumento exponencial do uso de automóveis para a locomoção, encontrar um lugar para estacionar, principalmente nos grandes centros urbanos, é uma atividade que se tem se tornado bastante difícil, e essa dificuldade, pode acabar gastando bastante tempo do motorista, o que causa estresse, e atrasos para os seus compromissos. Problemas de trânsito já existem há bastante tempo, e fazer com que pessoas não fiquem perdidas andando devagar procurando por lugares para estacionar ajudaria bastante o trânsito das cidades.

Além disso, o ato de procurar ficar dando a volta no quarteirão a procura de vaga, faz com que os motoristas gastem mais gasolina, recurso que atualmente no Brasil está bastante caro, ou seja, este aplicativo no final das contas ajudará os motoristas a economizarem dinheiro. Outro problema gerado pelas pessoas que ficam rodando a procura de vagas é a poluição, pois quanto mais tempo o carro fica ligado, mais monóxido

de carbono ele libera no ar, se o aplicativo tiver um alcance bastante grande, a quantidade de carros que gastarão menos tempo procurando vagas fará com que haja uma menor poluição do planeta.

1.3. Condições de Contorno.

Este trabalho, será executado dentro do laboratório de inteligência natural e artificial (LINA) e é baseado em uma ideia do professor Luciano Vieira Lima, entretanto que não havia sido executada ainda.

O trabalho se baseará em um questionário criado para fazer o estudo de mercado que será aplicado por mídia online, e através das respostas obtidas será desenvolvido o relatório de viabilidade do projeto, sendo que independentemente dos resultados, um protótipo funcional do aplicativo será desenvolvido, para trabalhar os conceitos de programação para aparelhos mobile, que é um dos focos do laboratório LINA.

1.4. Estrutura do Trabalho.

O trabalho foi desenvolvido para que tenha uma introdução explicativa onde serão tratados os objetivos do tema e como o trabalho será desenvolvido. Após ela será desenvolvido uma revisão teórica com todos os tópicos abordados no trabalho de conclusão de curso explicando os conhecimentos da área da engenharia e da área da administração que serão abordados.

Depois desta explicação o trabalho vai começara a tratar do desenvolvimento dos temas citados na revisão explicando como estes temas foram ou serão utilizados para que o projeto fosse executado, e o aplicativo mobile desenvolvido. Nesta existem 2 capítulos diferentes, o primeiro, no qual será explicado toda a parte administrativa e a utilização das ferramentas da administração para o desenvolvimento de uma empresa, e a checagem da viabilidade do produto que será desenvolvido, além de organizar toda a parte

de desenvolvimento do software. E a segunda parte que será contado como o aplicativo foi desenvolvido e quais as implementações futuras, além de mostrar detalhes que devem ser observados no futuro para estas implementações.

Por fim será criado uma conclusão que tem o objetivo de fechar o trabalho, trazer os pontos mais importantes de cada capítulo e falar dos aprendizados do projeto para o discente.

1.5. Estrutura do Aplicativo.

A ideia central do aplicativo está vinculado a dificuldade, das pessoas de encontrarem vagas para estacionarem seus carros principalmente nos grandes centros Urbanos, o que faz com que pessoas acabem perdendo bastante do seu tempo circulando envolta dos centros procurando as lugares para estacionar, sendo que seria muito mais produtivo um aplicativo reunisse as informações de onde existem vagas disponíveis, e com apenas um clique no celular o GPS forneceria uma rota para esta pessoa, até a vaga, além o aplicativo já marcar esta vaga para a primeira pessoa que tocou para que não haja uma competição pelo mesmo local de estacionamento.

Sabendo disso, foi pensado no monitoramento de 3 tipos de vagas:

- Vagas particulares ociosas em prédios ou moradias das pessoas, no qual o dono pode vender a vaga para outras pessoas estacionarem quando elas precisam ir em algum lugar próximo da moradia daquela pessoa.
- Vagas particulares de estacionamentos, registrando os estacionamentos da cidade, além de que criar um método de reserva de vaga, pelo aplicativo, para que a pessoa consiga reservar sua vaga em um estacionamento que geralmente fica lotado durante os dias, além de proporcionar a possibilidade do pagamento da vaga online.
- Vagas públicas nos centros das cidades, que serão monitoradas por câmeras públicas e/ou privadas, para saber se um determinado local é uma possível vaga, se ela está ocupada ou livre.

Tendo a ideia dos 3 tipos de vagas que o aplicativo irá monitorar, será explicado a lógica de como ele fará isso nos próximos tópicos.

1.5.1. Vagas Ociosas em prédios ou moradias

Para que o aplicativo consiga saber aonde estão estas vagas, é necessário que o usuário se cadastre no aplicativo com alguns dados, também cadastre o local aonde se encontra a sua vaga e os horários que ela estará disponível. A ideia aqui, é que ao cadastrar a vaga, ela fique aparecendo no gps para que as pessoas saibam que ela exista, e ao clicar nela, elas possam reservar esta vaga pelo aplicativo, e também conseguir o contato para se comunicar com a pessoa dona da vaga para que tudo possa ficar acertado para o momento em que o usuário for utilizar a vaga.

Para isso ser completamente eficiente, é necessário que o aplicativo também contenha um local para o cadastro de cartão de crédito, e conte com um sistema de pagamentos pelo aplicativo, visto que a pessoa ao reservar a vaga terá que efetuar o pagamento, para que caso está não compareça ao local no dia combinado, o dono da vaga não fique com ela bloqueada sem receber dinheiro. Além disso o dinheiro do pagamento só será liberado para o dono da vaga após a utilização dela, para que também não ocorra de o dono não querer emprestar a vaga dele para quem pagou por ela, por já ter recebido o dinheiro.

1.5.2. Vagas particulares de estacionamentos particulares.

A metodologia para a criação desta parte do aplicativo, é bastante semelhante a anterior, o dono do estacionamento faz o cadastro do estacionamento dele, e informa quantas vagas ele tem, sempre que entra um carro ou sai um carro ele deve colocar no aplicativo, para que ele informe a quantidade de vagas disponíveis. Os usuários podem ver em tempo real quantas vagas a no estacionamento, e podem reservar uma vaga para um determinado horário assim como feito no primeiro caso, o pagamento é feito adiantado, sendo que caso o usuário não compareça ao local no momento combinado ele não prejudica o dono do estabelecimento, e caso o dono do estabelecimento não deixe por algum motivo o usuário estacionar, o usuário pode informar o

aplicativo, fazendo com que o dono do estabelecimento não receba o dinheiro da reserva e o mesmo seja restituído ao usuário.

Em um modelo mais automatizado do software que não será criado neste primeiro momento, devido a sua complexidade, dono do estabelecimento pode instalar câmeras para filmar as suas vagas, e através de um algoritmo de Deep Learning que reconhece o que são as vagas, e se elas estão vazias ou ocupadas conseguirá atualizar o número de vagas disponíveis no software do aplicativo de forma automática, o que facilitará muito a vida do dono do estabelecimento, além de evitar erros de preenchimento de vagas errado.

1.5.3. Vagas Públicas.

Para o monitoramento das vagas públicas, seria necessário ter acesso as câmeras públicas que monitoram as ruas principalmente nos centros das cidades, e talvez colocar algumas câmeras para aumentar o número de câmeras no centro da cidade, pois neste caso quanto mais o número de câmeras maior o número de locais monitorados, portanto maior o número de vagas monitoradas. Para que essas câmeras conseguissem fazer o monitoramento, a imagem delas seria mandada para a nuvem e um servidor, com grande capacidade computacional, iria analisar nestas imagens através de um algoritmo de reconhecimento criado em Deep Learning, se existem vagas disponíveis ou não tais lugares, e passar estas informações para o aplicativo, atualizando sempre aonde se encontram as vagas disponíveis mostrando no mapa aonde elas se encontram.

Como estas vagas tem uma rotatividade muito grande, para esta parte do aplicativo, o próprio aplicativo tendo conexão à internet monitoraria todos os usuários, e caso houvesse mais de um clique na mesma vaga ele daria a vaga para a pessoa mais perto dela, e avisaria o outro usuário que a vaga já foi selecionada por outro usuário, evitando com que houvesse encontros inesperados, e que um usuário dirija até algum local atoa.

Esta parte do aplicativo, apesar de ser muito interessante, não será criada neste primeiro momento pela complexidade de programação envolvida, e também pela burocracia que haveria para conseguir acesso as imagens obtidas pelas câmeras públicas além do auto investimento que teria de ser feito caso fosse instalar mais câmeras para haver um monitoramento maior das vagas.

2. Fundamentação Teórica.

Nesta parte do trabalho será colocado os temas estudados para a o desenvolvimento do aplicativo como um produto vendável, então será explicado uma técnica para criação de um produto ou negócio que foi aplicada no desenvolvimento do projeto, além de explicar alguns princípios básicos e avançados de programação que serão utilizados no decorrer do trabalho, e também que devem ser utilizados para a melhoria do aplicativo como citado na parte de introdução do projeto.

2.1. Modelo de negócio Canvas.

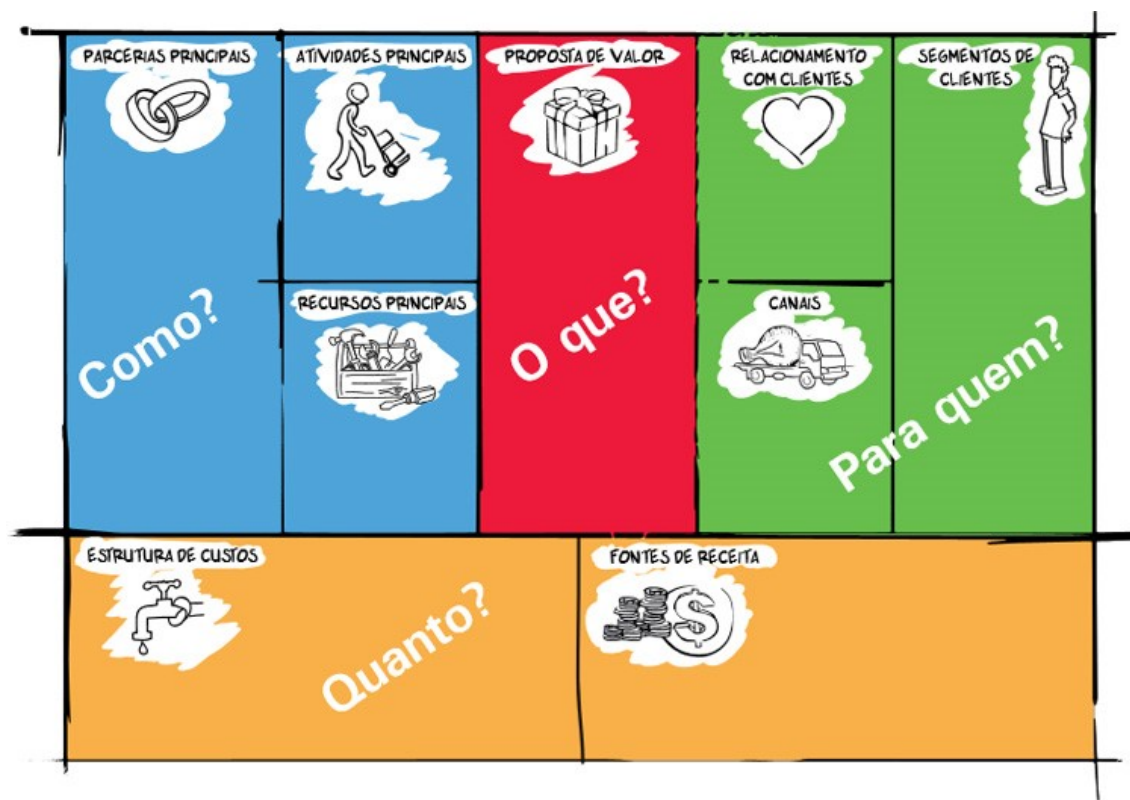
O modelo de negócio canvas é uma ferramenta moderna utilizada para a criação de um modelo de negócio atual, bastante simples, que visa que a empresa consiga definir os principais pontos de toda a empresa, e como funcionará a sua geração de renda.

Este modelo se baseia na criação de um quadro, que facilita a visualização de todas as partes da empresa, definindo bem os stakeholders e todo o processo de geração de valor, desde os produtos da empresa, até as formas de comunicação com o cliente, tentando responder algumas perguntas base, que são:

- O que? (Proposta de valor, tem a ver com o produto ou serviço)
- Para quem? (Clientes, e meios de entrega e comunicação com o cliente)
- Como? (atividades, recursos e parcerias, para a entrega do valor)
- Quanto custa? (quanto custa para a proposta ser criada, e como ela será monetizada)

Além disso como já foi citado anteriormente, é bastante comum preencher o quadro do modelo canvas com post-its, e normalmente o quadro é preenchido com post-its coloridos, sendo que cada cor representa, um tipo de cliente do modelo de negócio. O exemplo de como é uma tabela do modelo canvas, é mostrado na figura 2.1.

Figura 2.1 - Quadro que representa o modelo canvas



Fonte: <http://www.sebrae.com.br/sites/PortalSebrae/bis/quadro-demodelo-de-negocios-para-criar-recrutar-e-novar,a6df0cc7f4217410VgnVCM2000003c74010aRCRD>

Para que seja mais fácil de entender explicarei cada parte deste quadro na continuação deste texto, separando pelas perguntas já citadas anteriormente.

2.1.1. O que?

Esta parte do modelo canvas tem como a principal função, definir quais são os produtos, ou serviços que a empresa vai entregar para os seus clientes, e como estes produtos geram valor para os seus clientes. Esta parte do modelo canvas só tem 1 quadro, e este deve ser o segundo quadro a ser preenchido para que o modelo seja usado corretamente. O quadro que representa esta parte, é o quadro da proposta de valor, nele, deve ser escrito, ou colocado vários pous-its, descrevendo de forma extremamente sucinta e direta quais os produtos e serviços oferecidos pela empresa.

2.1.2. Para Quem?

Esta parte do modelo Canvas é composta por 3 quadros (segmentos), sendo eles, o segmento de cliente, relacionamento com os clientes e o segmento de canais.

- Segmento de clientes: Este quadro é o primeiro quadro que deve ser preenchido, no momento de se criar o modelo de negócios, pois recentemente, as empresas descobriram de forma muito lógica, que toda a atenção e criação de produtos e serviços deve ser focada para atender o seu consumidor, então os modelos de negócio atuais, tem foco total no cliente. Desta forma, para que haja um melhor aproveitamento do modelo canvas, e da criação de valor gerada pelo seu produto ou serviço, este deve ser o primeiro quadro a ser preenchido.
- Relacionamento com o Cliente: Neste quadro que deve ser preenchido em 3 ou quarto lugar, a empresa irá estabelecer as formas de comunicação que ela terá para que seu cliente consiga se comunicar com a empresa, portanto, aqui serão estabelecidos se a empresa terá um telefone de contato, uma central de suporte, um e-mail de reclamações, um e-mail de recomendações, pessoas próprias da empresa que devem entrar em contato com os cliente para pegar opinião etc.
- Canais: Este quadro assim como o anterior deve ser o 3 ou quarto quadro a ser preenchido, e ele irá estabelecer as formas de comunicação da empresa com o cliente, e as formas de entrega de produto ou serviço caso necessário. Portanto, neste segmento será estabelecido as formas de propaganda da empresa, como serão criadas as redes sociais da empresa, se ela terá de ter uma transportadora como parceira, se ela terá de comprar transporte para entregar seus produtos ou serviços etc.

Lembrando sempre que cada quadro do modelo canvas deve ser preenchido com frases sucintas e diretas, de preferência com adesivos que possam ser removidos, pois caso seja necessário a troca de algo que foi definido no começo para o melhor atendimento do cliente, isto seja possível de ser feito.

2.1.3. Como?

Nesta parte do modelo canvas será estabelecido os métodos de criação e desenvolvimento do produto ou atividade, onde serão levantadas as atividades que devem ser executadas pela empresa para que o produto consiga ser desenvolvido, vendido, e ainda ter o suporte necessário para o seu funcionamento, e devem ser o sexto sétimo e oitavo quadros a serem preenchidos, não importando tanto a ordem entre eles de qual o usuário irá preencher primeiro, mas normalmente o quadro de atividades principais é preenchido primeiro que os outros.

- Atividades Principais: neste segmento serão destrinchadas todas as atividades desenvolvidas pela empresa, para que o produto ou atividade possa ser desenvolvida da melhor maneira possível atendendo os quadros da criação de valor, canais e relacionamento com o cliente.
- Recursos Principais: Neste quadro são apontados todos os recursos necessários para a execução do projeto, sendo estes, tanto matéria prima, quanto funcionários que deverão trabalhar em áreas específicas do projeto.
- Parcerias Principais: Neste quadro deve-se destrinchar as parcerias necessárias que a empresa deve implementar, para garantir um melhor preço do seu produto, e ou serviço, além de conseguir entregar o produto de forma mais rápida e melhor acaba, com a ajuda de seus parceiros.

2.1.4. Quanto Custa?

Nesta parte do modelo de negócios serão preenchidos e descritos as fontes de receita do projeto, destrinchando elas em pequenas frases objetivas e simples de serem lembradas, que deve ser o 5 quadro a ser preenchido, e também a parte de estruturas de

custos, que são os custos inerentes dos processos da empresa (todos eles) que a empresa gastará mensalmente para funcionar, após feito isso, é interessante ter uma ideia de quanto serão estes custos, por tanto, deve ser feito um levantamento de custos, para entender o quanto é necessário ser gerado pela receita, para que a empresa consiga se manter. O quadro dos custos, sempre é o último quadro a ser preenchido.

Após preenchido todos estes quadros, o empreendedor saberá com facilidade tudo que deve ser feito para que os processos funcionem, e caso seja necessário alterar algum processo, após ter estruturado o projeto, e bastante simples, do modelo de negócio ser alterado; e esta é a grande vantagem do modelo canvas, ele é bastante rápido de ser criado e altamente simples de ser modificado.

2.2. Pesquisa de mercado

A pesquisa de mercado é uma atividade feita pelas empresas normalmente desenvolvida para adquirir dados à cerca de seus clientes, para que a empresa consiga desenvolver de forma mais efetiva os seus serviços e produtos atendendo de forma mais eficiente, um determinado segmento do mercado. A pesquisa de mercado, também é uma atividade criada para avaliar a viabilidade de um novo produto, que será lançado pela empresa, e será criado neste trabalho de conclusão de curso, uma pesquisa de mercado para avaliar a necessidade dos clientes para um novo produto, logo será falado mais deste tipo de pesquisa de mercado.

Existem várias formas diferentes de se utilizar desta ferramenta, e estas maneiras, ganham nomes específicos na literatura, portanto, será explicado os tipos de pesquisa de mercado existentes, e quais as suas diferenças na prática.

- Pesquisa de mercado qualitativa: Normalmente utilizado, quando não se deseja analisar um grande número de pessoas, se baseia na aquisição de dados, sem se preocupar muito com a parte estatística, e foca mais no conteúdo que pode ser gerado pela resposta da pesquisa, logo esta é criada geralmente com questões abertas, muitas vezes de cunho subjetivo, e que requerem que a pessoa que está respondendo pense um pouco mais.
- Pesquisa de mercado quantitativa: Diferente do modelo anterior é completamente focada em cálculos estatísticos, visa atingir um número maior de pessoas, e tem

como objetivo responder algumas dúvidas pré-definidas pelo grupo que criou a pesquisa. Usa como forma de aquisição de dados, normalmente questões objetivas, muitas vezes criando sistemas de notas para avaliações de processos ou produtos da empresa.

- Técnica de observação: Este tipo de pesquisa é bastante simples, pois ela é feita por um grupo de pessoas, que observa os seus clientes naturalmente em alguma atividade específica que este grupo está avaliando, e de forma subjetiva o grupo tira informações sobre estes processos. Este modelo pode ser aplicado tanto na própria empresa, quanto em empresas concorrente, para tentar entender como os concorrentes desenvolvem determinados processos.
- Técnicas experimentais: É criado um ambiente artificial para entender como ocorreria a implementação de um processo ou produto. Neste ambiente pelo menos uma das variáveis é manipulada, pelo grupo pesquisador, para entender com as outras se comportam.
- Pesquisa A/B: O pesquisador segmenta sua base (produtos, lojas, processos etc.) e em cada grupo ele aplica um experimento diferente e avalia ao longo do tempo. Com o passar do tempo um dos grupos terá um resultado mais satisfatório que o outro, então, o pesquisador pega este modelo de experimento aplicado e utiliza ele em toda a empresa.

Entendendo um pouco dos tipos de pesquisa de mercado, deve-se citar, que para o segmento deste trabalho, será aplicado uma pesquisa de mercado de cunho quantitativo, e que terá como principal objetivo avaliar a viabilidade de uma nova atividade da empresa (no caso avaliar a viabilidade da empresa, visto que a mesma não existe ainda.)

2.3. Matriz GUT

A matriz GUT é uma ferramenta fundamental da qualidade total, utilizada para priorizar atividades e descobrir a ordem que as atividades devem seguir para o melhor andamento dos processos de uma empresa, ou simplesmente para que uma meta individual seja atingida. Esta matriz carrega o nome GUT porque ela analisa 3 requisitos básicos, gravidade, urgência e tendência, e dá uma nota de 1 a 5 para estes quesitos, sendo que a nota final da atividade analisada é a soma dos números dados. A figura 2.2 mostra como é uma matriz GUT.

faz com que a situação piore no decorrer do tempo, as notas deste requisito variam de não vai piorar (nota 1) até vai piorar rapidamente (nota 5).

2.4. Paradigma computacional.

Parafraseando o site: <<https://www.significados.com.br/paradigma/>> A palavra paradigma significa: modelo ou padrão, a ser seguido em uma determinada situação ou para executar uma determinada tarefa. O paradigma são normas orientadoras que limitam como o indivíduo deve agir dentro destes limites. Para a programação, é muito importante saber os paradigmas existentes, e qual o melhor paradigma escolher para executar uma determinada tarefa, pois, ele faz muita diferença na hora de programar o programa desejado, podendo deixar a programação mais fácil e fazendo grande diferença no resultado do programa.

Os tipos de paradigma presentes na programação são diversos sendo algumas que algumas linguagens de programação, como o C++, suportam múltiplos tipos de paradigma. Mas serão citados aqui apenas quatro, pois são os que estão sendo considerados para o desenvolvimento do aplicativo de treinamento de pronúncia do inglês, são estes: programação orientada a objetos, programação imperativa, visual e genérica.

2.4.1. Programação orientada a objetos (POO):

Este tipo de linguagem de programação tem como base a interação de várias unidades chamadas de objetos, estes objetos além de conseguirem interagir com outros, também podem inter-relacionar-se com si mesmo, alterando suas próprias instancias. Além disso normalmente a POO geralmente trabalha com classes, e os objetos são instancias destas classes, e também as classes definem como este objeto pode se comportar durante a programação. São exemplos de linguagem de programação que utilizam este paradigma: java, lua e C++, sendo que lua e C++ utilizam múltiplos paradigmas, não sendo totalmente restritas a este apresentado.

2.4.2. Programação Imperativa:

Este tipo de paradigma se baseia na criação de ações, enunciados e comandos que o computador deve executar em sequência e que alteram o estado e as variáveis do programa. Assim como o modo verbal imperativo da língua portuguesa, a programação imperativa se baseia em dar ordens para o computador executar, isto faz com que este paradigma seja bastante simples, pois o programador deve abstrair apenas as ordens que ele dará ao computador e a sequência que estas devem seguir. A linguagem imperativa trabalha muito com o armazenamento de dados de variáveis em posições definidas de memória que são alterados pelos comandos da linguagem de forma direta ou indireta. A programação imperativa, também pode ser chamada de programação procedural e é o paradigma de programação mais antigo existente, pois ele também é um dos modelos mais fáceis de se transformar para a linguagem de máquina (compilar). São exemplos de linguagens de programação que utilizam este paradigma: C++, C, Fortran, lua.

2.3.4 Programação visual:

A programação visual, é o modelo mais simples de programação que existe, pois ela tem um nível alto de abstração, sendo que a programação se baseia em mover e encaixar blocos que contêm funções, tarefas, variáveis e iterações. Como a programação é feita por bloquinhos fixo significam algo, fica muito mais fácil de entender o que o programa está fazendo, portanto, fica muito mais fácil de programar. Entretanto, a programação visual costuma ser um pouco limitada, visto que, caso não exista um bloquinho que faz uma atividade que o programador deseja, ele terá de encontrar outro método de fazer aquela atividade com as funções e iterações prontas que já existem na linguagem utilizada. Algumas linguagens visuais, dão a possibilidade de o programador criar uma função que não exista nela através de outra linguagem escrita, como é o exemplo do AppInventor, que permite que o programador crie bloquinhos para ele utilizando-se da linguagem java.

2.3.5 Programação Genérica:

Este tipo de paradigma se baseia em uma gramática estendida que permite que ela se adapte para as variáveis definidas na instância do algoritmo, isto faz com que a semântica deste tipo de linguagem seja genérica, e vários parâmetros possam assumir diferentes tipos de dados desde que certas regras sejam mantidas, como subtipos e assinaturas. Por exemplo a declaração de uma lista neste tipo de programação seria: List(T), no qual T seria um tipo qualquer de variável, visto que uma lista pode ter variáveis como números inteiros, ou até mesmo strings, como por exemplos nomes de produtos para serem comprados no supermercado. São exemplos de linguagens de programação que se utilizam deste paradigma: C++ e lua.

2.4 Escolha da Plataforma de programação Mobile.

2.4.1 Java:

A linguagem Java criada na década de 90 pela Sun Microsystems que posteriormente foi comprada pela Oracle é uma codificação orientada a objetos, que pode ser utilizada no desenvolvimento de aplicativos móveis. Por se tratar de uma linguagem compilada para uma máquina virtual a JVM (Java Virtual Machine), os códigos de Java possuem uma boa portabilidade em qualquer tipo de computador, o que foi revolucionário na época de sua criação, e fez com que esta programação fosse uma das mais utilizadas até hoje. Como a linguagem Java nos seus primórdio não foi criada para o desenvolvimento de aplicativos móveis, desenvolver aplicativos com este tipo de código, acaba sendo uma tarefa bastante trabalhosa, precisando de extensas linhas de código para se fazer coisas muito simples para que o aparelho do celular aceite, e para que os sistemas operacionais de cada celular (Android ou IOS) aceitem que o aplicativo funcione de formas iguais, mesmo com a criação do aplicativo android studio e algumas bibliotecas que ajudam a programação, criar um aplicativo em java ainda exige bastante empenho e muitas linhas de código, portanto a opção de se usar java foi descartada.

2.4.2 C++:

C++ é uma variação da linguagem C, ela foi desenvolvida no começo da década de 80, e era chamada de “C with classes” (C com classes). Este tipo de código pode ser utilizado para os mais diversos paradigmas, sendo classificada como uma linguagem multiuso, extremamente famosa com uma variedade muito grande de bibliotecas e funções, o que faz com que ela possa ser utilizada para a criação de aplicativos mobile, entretanto, igual a linguagem java, o C++ não foi desenvolvido pensando na programação de programas mobile, então para criar um aplicativo para mobile utilizando esta linguagem é necessário de bastante trabalho, e inúmeras linhas de código para que sejam executadas tarefas bastantes simples para as diversas plataformas de celulares e para os 2 sistemas operacionais existentes (Android e IOS), por isso a programação em C++ também foi dispensada.

2.4.3 Corona SDK:

O Framework Corona SDK foi desenvolvido em 2009 pela corona Labs, feito especificamente para o desenvolvimento de aplicativos móveis. O Programa se utiliza da linguagem e programação em lua, uma linguagem criada sobre a linguagem C++ para facilitar a escrita e aplicação na parte gráfica, a figura 2.3 mostrada abaixo mostra como é o framework do Corona SDK.

Figura.2.3 - Framework do programa Corona SDK



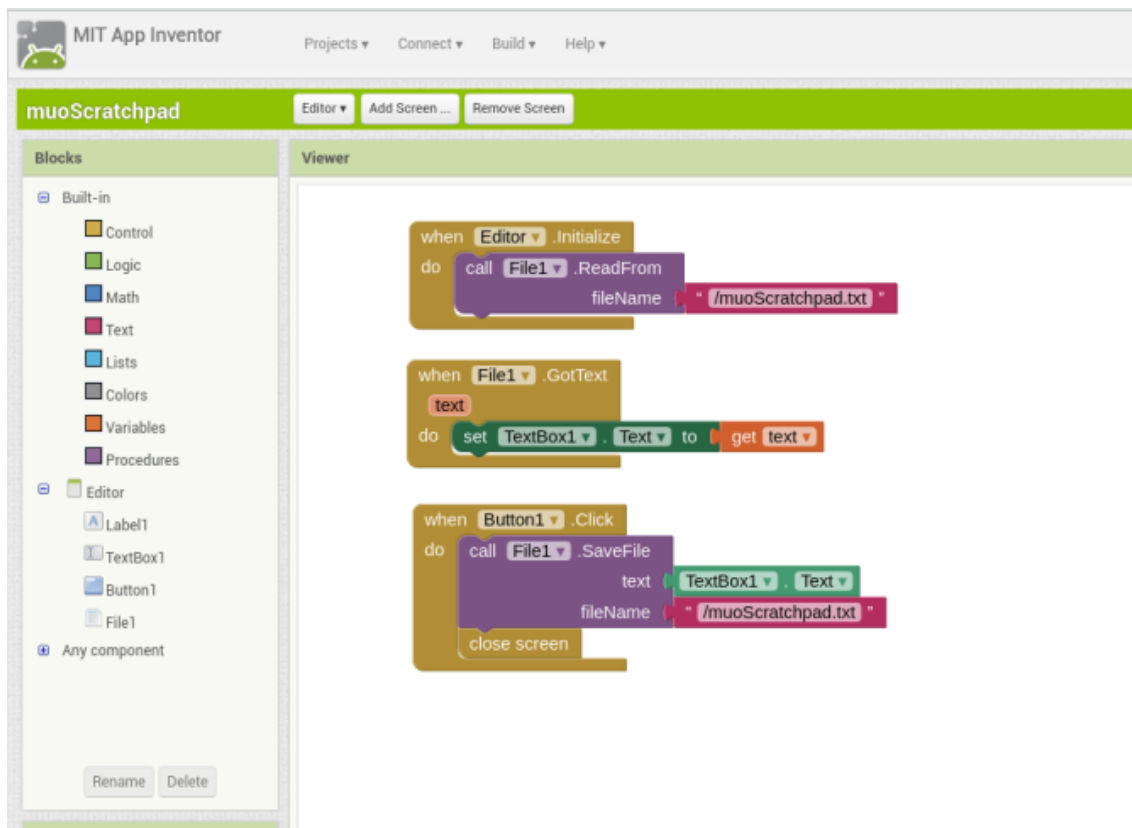
Fonte :< <http://coronalabs.com>> foto copiada dia 31/10/2018

O corona SDK, diferente dos outros programas foi criado para desenvolvimento em plataformas mobile, e plataformas como smart tvs. Apesar de ser bastante conhecido e utilizado para o desenvolvimento de jogos mobile 2D, o Corona SDK também pode ser utilizado para a criação de vários tipos de aplicativos diferentes que podem ser dos mais simples, como um cronometro, até aplicações bastante complexas que requerem reconhecimento de imagem etc.

2.4.4 APP Inventor.

APP Inventor foi um framework criado pela Google e atualmente é mantido pelo Massachusetts Institute of Technology (MIT) para o desenvolvimento de programas para Android. O APP Inventor tem uma linguagem de mais alto nível baseada na linguagem Java, no qual ele se baseia em uma programação por blocos, no qual, os usuários arrasta bloquinhos e os liga para o desenvolvimento de um aplicativo, que apesar de parecer um modelo de programação bastante simplista, ele é extremamente completo, fazendo com que usuários possam criar aplicativos bastante complexos como de reconhecimento de voz ou aplicativos de interação GPS etc., a figura 2.4 mostra o framework do APP inventor.

Figura.2.4 - Framework do APP inventor



Fonte: <<https://static.makeuseof.com/wp-content/uploads/2017/08/android-create-app-appinventor-screen2-blocks.png>> Copiada no dia 31/10/2018

Como pode-se ver na figura, o método de trabalho do app inventor é bastante simples, e o mais importante, é que este método já possui várias APIs da Google o que faz com que muitos aplicativos uteis consigam ser produzidos dentro do aplicativo. Entretanto, o App Inventor, tem uma gama limitada de ferramentas prontas, e ele até possui uma parte para o desenvolvimento de novas ferramentas, entretanto para utilizar esta parte é necessário saber programação em Java.

2.3.5 Plataforma Escolhida:

Pelo fato de uma programação mais simples, totalmente voltada para o desenvolvimento mobile, e com uma ampla possibilidade de uso das ferramentas, para este projeto será utilizado a plataforma Corona SDK, e a linguagem de programação lua, uma linguagem orientada a objetos, com um grau alto de abstração, porém com um alto range de atuação.

2.5 Deep Learning.

A ferramenta de deep learning não será utilizada na construção deste aplicativo, entretanto ela é de uma tremenda importância para a otimização do aplicativo, e para conseguir encontrar vagas em locais públicos, pois este tipo de algoritmo é o mais eficiente atualmente, para o reconhecimento de imagem, e através deste algoritmo seria possível fazer, com que, um software ao receber as imagens das câmeras públicas consiga identificar, se existem vagas livres ou ocupadas, e este passaria esta informação para o aplicativo.

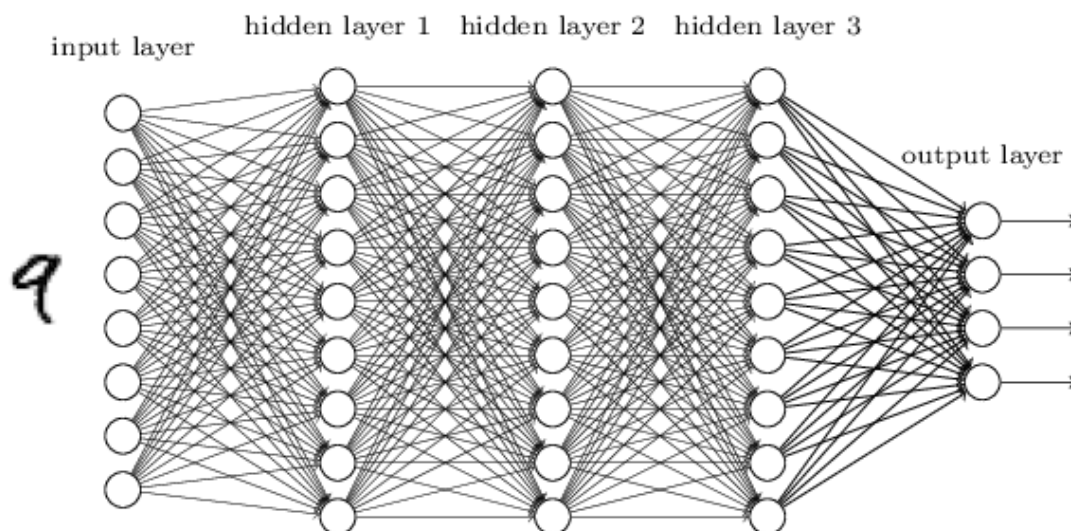
I. Conceito:

O deep learning é uma técnica de programação criada na década de 60, que se baseia nos princípios de tentar imitar o funcionamento dos neurônios humanos, para fazer com que as máquinas consigam ter um processo de abstração de alto nível e apreender uma informação, padrão ou atividade, e a mesma seja executada da maneira como ela aprendeu.

A técnica se baseia em um algoritmo multicamada no qual as camadas são criadas por células (números) que passam a informação para as camadas seguintes na forma de uma função matemática, nestas funções a entrada e a saída são conhecidas, ou seja... a camada de entrada ou input layer e a camada de saída ou outputlayer são conhecidas, e as camadas intermediárias não são conhecidas e ganham o nome de hidden layers. A figura

2.5 mostra como é representado uma estrutura de rede neural do algoritmo de deep learning.

Figura 2.5 - Estrutura que representa uma rede neural de um algoritmo de Deep Learning



Fonte adaptado de: <https://chsasank.github.io/deep-learning-crash-course-2.html>

Este modelo de algoritmo é baseado em um método de treino no qual é apresentado ao computador algo que ele tenha que reconhecer e a saída esperada para essa entrada, e a partir de mostrar diversas entradas e treinar o computador a ele executar uma tarefa específica, ou identificar uma coisa específica para esse dado mostrado e repetir isso diversas vezes até que ele consiga fazer as atividades sem que seja mostrado para a máquina a resposta esperada.

O deep learning é um algoritmo muito utilizado atualmente para reconhecimento de imagem e consegue ter uma taxa de acerto em suas atividades muito alta, chegando muito próximo de 100% se ele for bem treinado, e bem desenvolvido. Por conta disso este modelo foi citado aqui, pois ele pode ser utilizado nas implementações futuras, quando o propósito seria encontrar vagas em locais públicos, como já foi citado anteriormente no tópico 1.5

Capítulo 3

3. Criação do produto.

3.1. Criação do modelo de negócio

O modelo de negócio utilizado neste projeto, foi o modelo canvas, visto que ele é bastante simples de ser executado, e traz um excelente resultado para o projeto, visto que este modelo é um dos mais utilizados pelas startups atualmente.

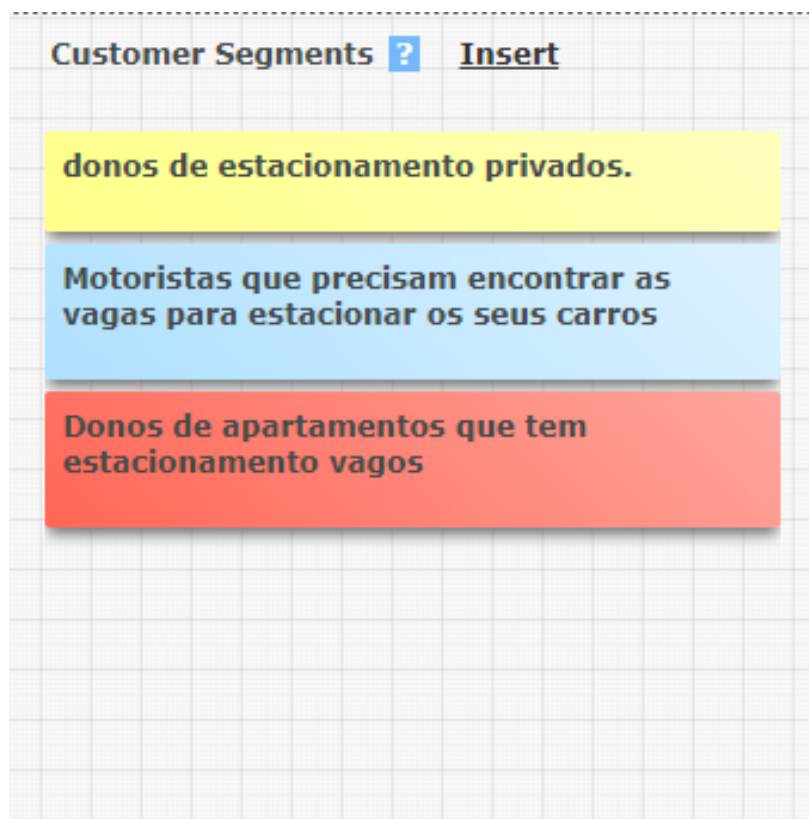
Portanto neste capítulo será respondido as perguntas do modelo canvas como os tópicos deste segmento do trabalho, além disso foi utilizado o a ferramenta do site Canvanizer.com, para criar o modelo de negócio.

3.1.1 *Para quem?*

Para responder essa pergunta basta apenas preencher o segmento de clientes, já que este é o segmento mais importante, como dito no tópico 2.1.

O preenchimento deste segmento, foram considerados 3 clientes potenciais, neste primeiro momento, não foi restrito a idade dos clientes, pois isso será aferido na parte de pesquisa de mercado, estes clientes são: Donos de estacionamento privado, Motoristas que precisam encontrar vagas, donos de apartamentos que tem estacionamentos vagos. O segmento preenchido no canvanizer é na figura 3.1.

Figura.3.1 - Segmento de clientes preenchido no canvanizer



Fonte de criação própria

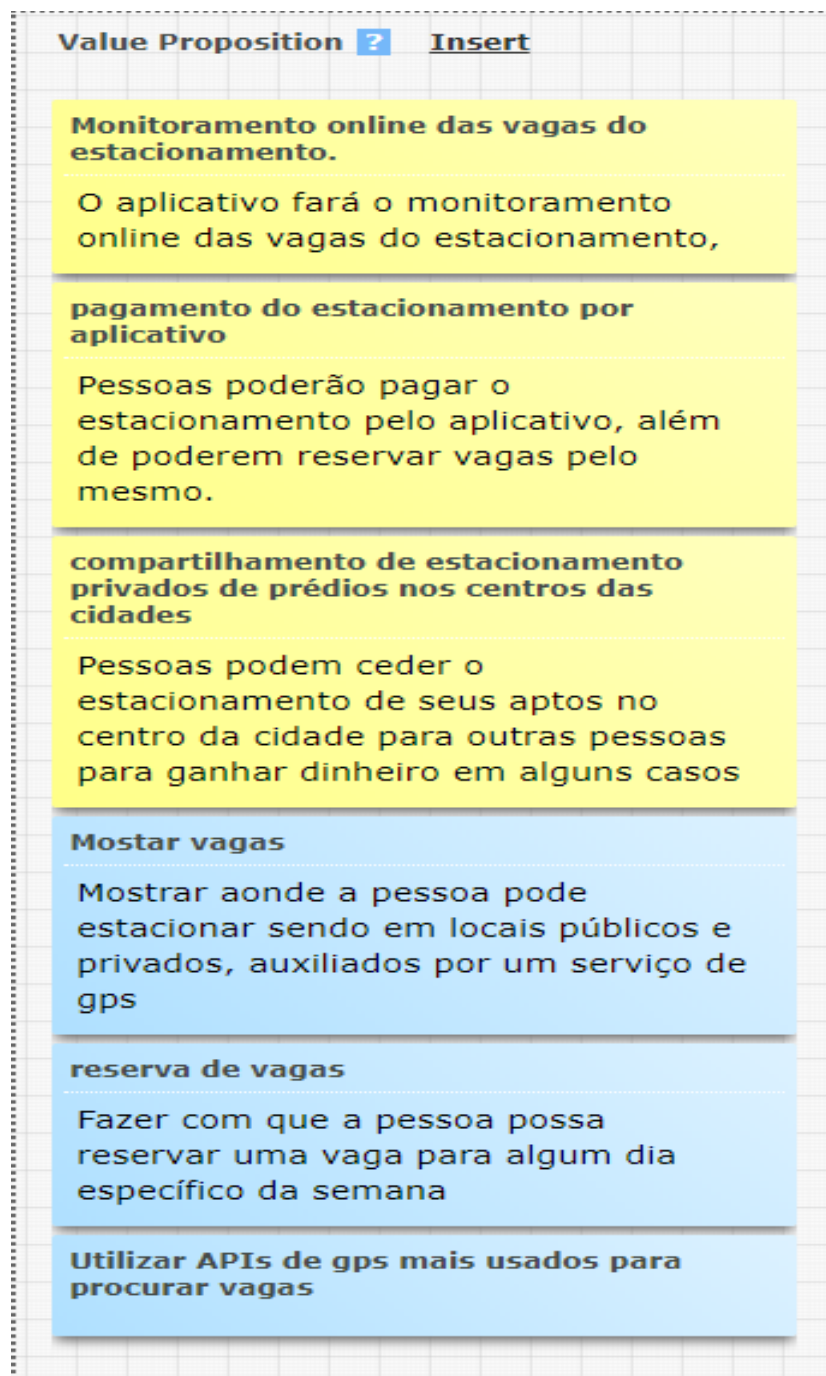
Estes 3 clientes foram escolhidos, por conta do modo como foi pensado o aplicativo, visto que o protótipo do aplicativo trabalhará para ajudar um motorista a encontrar vagas privadas disponíveis próximo ao seu destino, logo o aplicativo precisa saber quem são as pessoas que fornecem estas vagas, portanto, tanto as pessoas que fornecem as vagas, quanto os motoristas serão considerados usuários do produto, por isso foram listados nesta parte.

3.1.2 O que e para quem?

Nesta parte do modelo canvas, são preenchidos 3 quadros, como já citado na parte no capítulo 2, respeitando a ordem de preenchimento do modelo canvas, primeiramente foram levantadas as possíveis propostas de valor que o produto tem a oferecer para os clientes. A figura 3.2 mostra como o quadro da proposta de valor foi preenchido, é

bastante importante citar, que como o cliente dono de estacionamento privados, e donos de apartamentos que tem estacionamentos vagos, serão considerados com a mesma cor aqui (amarelo), visto que, o valor criado para eles é igual.

Figura.3.2 - Segmento da proposta de valor

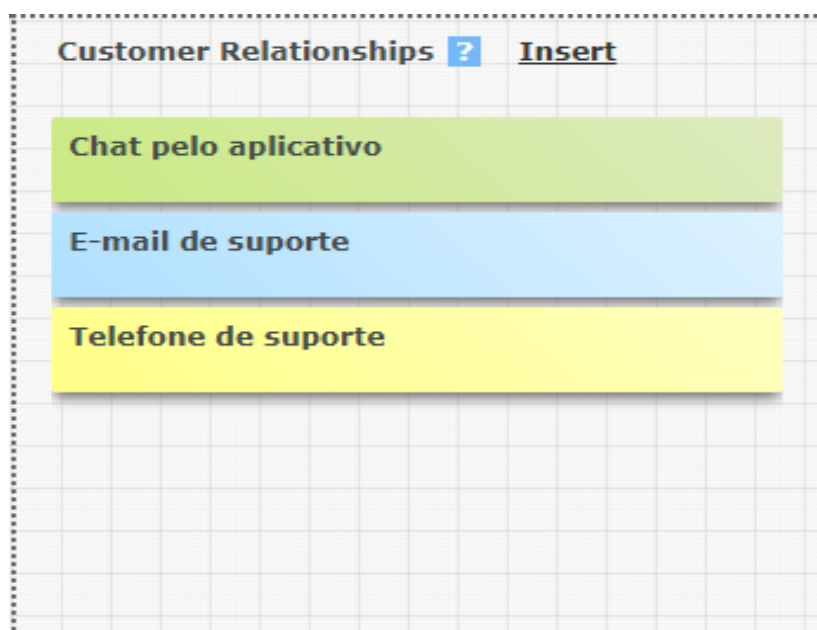


Fonte: criação própria.

Como pode ser visto na figura 3.2, ainda é considerado para o aplicativo completo encontrar vagas em locais públicos, além disso, também fica evidente o porquê de ter sido usado o modelo canvas, pois ele é bastante fácil de se utiliza-lo, ele é muito claro nas suas propostas e fácil de entender.

Após preenchido a parte de criação de valor, foi preenchida a parte de relação com o cliente, nesta parte, foi proposto que o cliente conseguirá entrar em contato com a empresa por 3 meios, um chat de suporte do aplicativo, além de existir também um e-mail de suporte, e também um telefone de suporte, no qual o telefone de suporte, por ser normalmente medidas mais urgentes, será utilizado normalmente pelos donos de estacionamento, e o e-mail será utilizado pelo motoristas. A figura 3.3. mostra como foi preenchido o segmento de relacionamento com o cliente do modelo canvas.

Figura.3.3 - Segmento de relacionamento com o cliente.

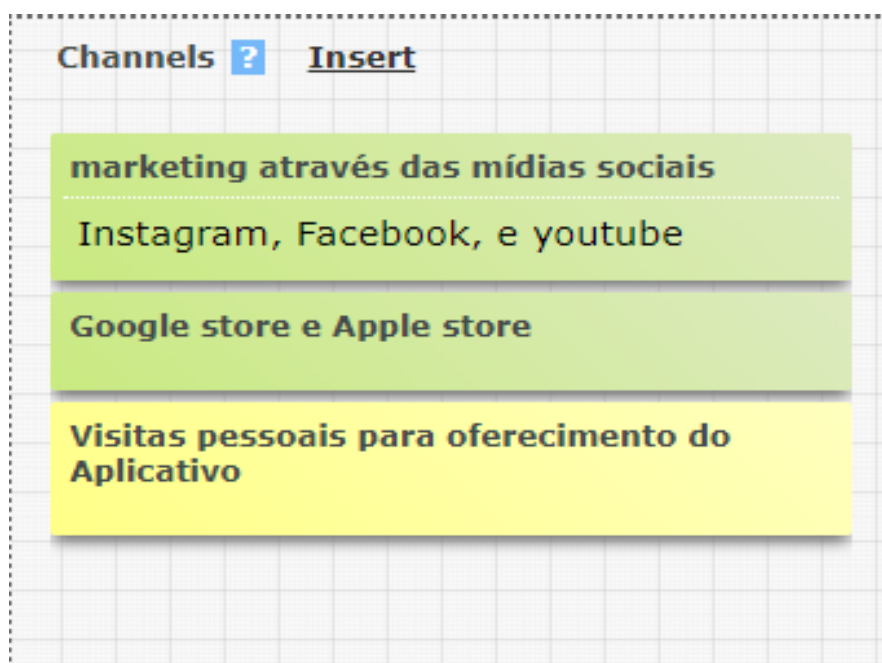


Fonte: Criação própria

Por último foi preenchido o segmento de canais, nos quais a empresa se comunicaria com o cliente, e os canais encontrados para contatar o cliente, fazendo propagandas, e disponibilização do produto foram: Marketing através de mídias sociais, tais como Facebook Instagram, e youtube; Google Store e Apple store, nos quais o aplicativo poderá ser baixado; visitas pessoais a estacionamentos para o oferecimento do

serviço. Sabendo os três tipos de canais de contato com o cliente, fica claro as áreas que terão de ser trabalhadas, dentro da empresa, para sustentar estes canais. Apesar do aplicativo ser totalmente digital, o último tipo de canal sugerido (visitas pessoais) existe pelo fato de a maioria dos estacionamentos privados no Brasil ainda serem bastante antiquados, visto que a maioria deles nem passa cartão, e os pagamentos ainda são todos feitos em dinheiro, logo é fácil de prever que os esforços para vender este produto para estes estabelecimentos vai precisar ter um esforço maior. A figura 3.4. mostra como foi preenchido o modelo canvas no site canvanizer.

Figura.3.4 - Como foi preenchido o segmento canais



Fonte: Criação Própria

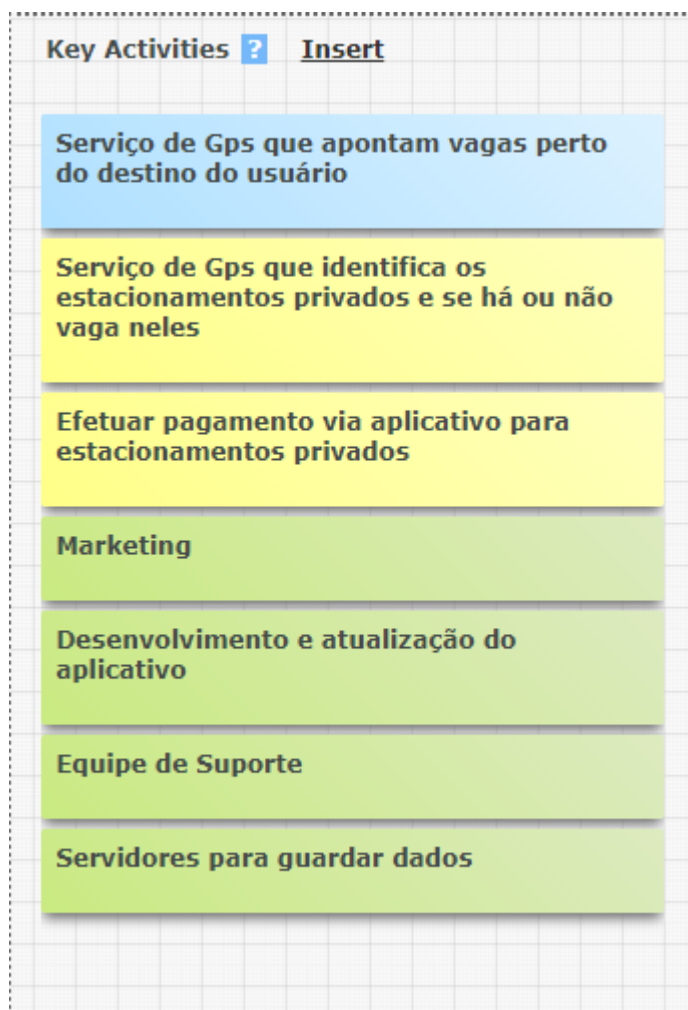
3.1.3 Como?

Dando continuidade ao preenchimento do modelo canvas, um pouco fora de ordem, pois como já explicado no capítulo 2, o modelo canvas é preenchido da direita para esquerda, logo o segmento de receita deve ser preenchido antes dos três segmentos que respondem à pergunta “como”. Esta ordem proposta pelo modelo foi seguida no momento do preenchimento, entretanto, para ficar

mais fácil de mostrar o preenchimento através da resposta das perguntas, ele será mostrado no próximo tópico.

Para que seja respondida à pergunta “Como”, devem ser preenchidos três segmentos, sendo eles: o segmento de atividades chave, recursos chave e parcerias. O primeiro quadro a ser preenchido foi o de atividades chave, nele foram evidenciados todos os trabalhos que devem ser executados além das funções que devem estar operantes para que o aplicativo consiga funcionar de forma adequada para os usuários. As ações levantadas que a empresa deve ter para que o aplicativo consiga funcionar são, ações de marketing em mídias, e suporte ao cliente; além disso para o bom funcionamento do aplicativo é necessário que estejam operantes as funções de mostrar vagas privadas e públicas (lembrando que o canvas do projeto ainda considera o aplicativo completo e não o protótipo), além da função de pagar o estacionamento via aplicativo e também um servidor funcional que consiga guardar os dados dos clientes para o melhor funcionamento do software. A figura 3.5 mostra como foi preenchido o segmento de atividades chave.

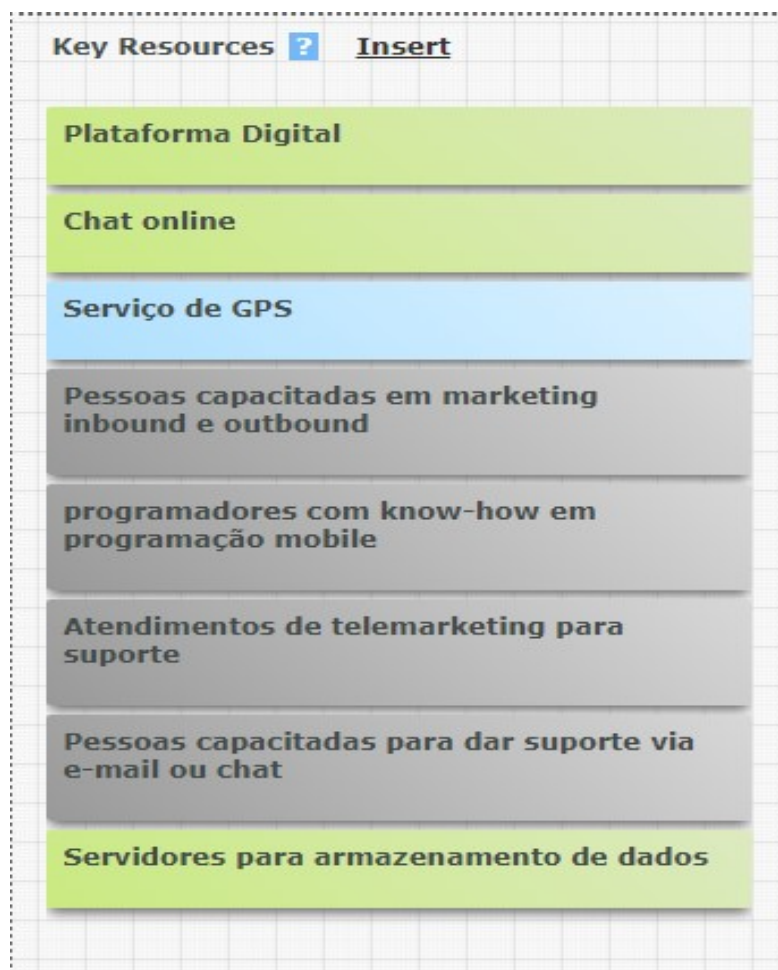
Figura.3.5 - Como foi preenchido o segmento de atividades chave.



Fonte: Criação Própria

Dando continuidade, ao processo de preenchimento do modelo canvas, o próximo segmento a ser contemplado foi o de recursos chaves. A figura 3.6 mostra a secção de recursos chave.

Figura.3.6 - Preenchimento do segmento de recursos chave

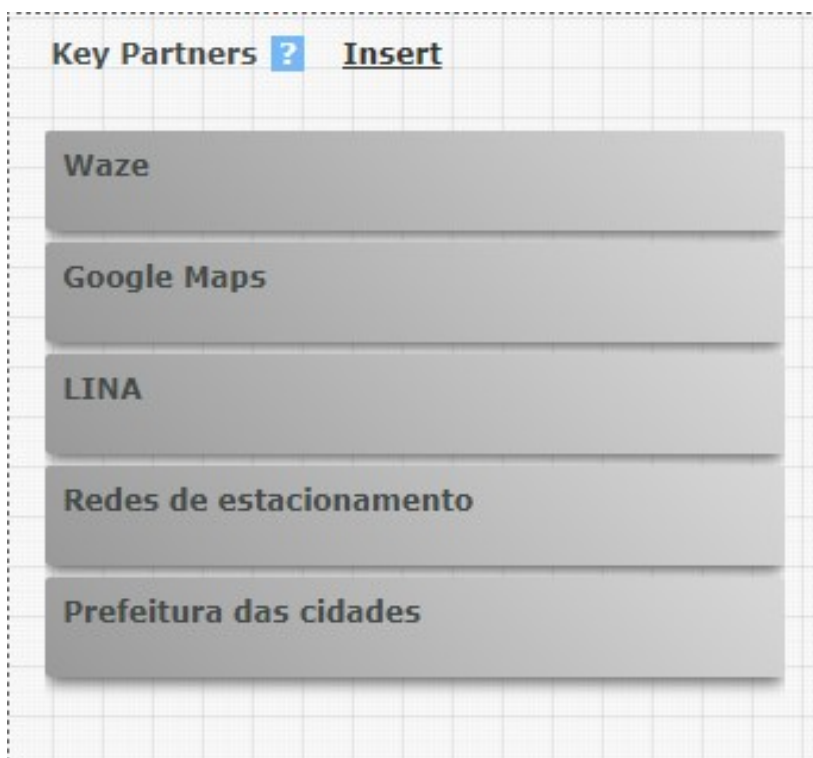


Fonte de criação própria

Para completar esta parte do projeto, foram considerados os seguintes recursos: Plataforma digital, Chat online via aplicativo, Serviço de GPS, Pessoas capacitadas em marketing inbound e outbound, programadores com know-how em programação mobile, atendimento de telemarketing para suporte, pessoas capacitadas para dar suporte via chat ou via e-mail e servidores para o armazenamento de dados. Estes recursos foram pensando em cima de tudo aquilo que já foi citado anteriormente no decorrer do modelo canvas, principalmente para atender os segmentos de “geração de valor”, “canais” e “relacionamento com o cliente”.

Tendo esta parte do modelo completa, então será preenchido o último segmento da pergunta “como” do projeto, que se trata da parte de parcerias. A figura 3.7 mostra como esta parte do quadro foi preenchida.

Figura.3.7 - Preenchimento do segmento de parceiros chave.



Fonte: Criação própria

Nesta parte do modelo canvas, foram considerados parceiros, que já são grandes referências na parte de gps para celulares no mercado atual, além de claro procurar por grandes estacionamentos parceiros, que facilitariam além da divulgação do aplicativo, também a venda dele; também foi considerado as prefeituras da cidade para que seja possível ter acesso as câmeras públicas visando o monitoramento de vagas em locais públicos, como será explicado no final deste trabalho; e por fim o Laboratório no qual este projeto está sendo desenvolvido, para que o mesmo possa ter melhorias com o tempo além de uma equipe qualificada para propor melhorias ao aplicativo.

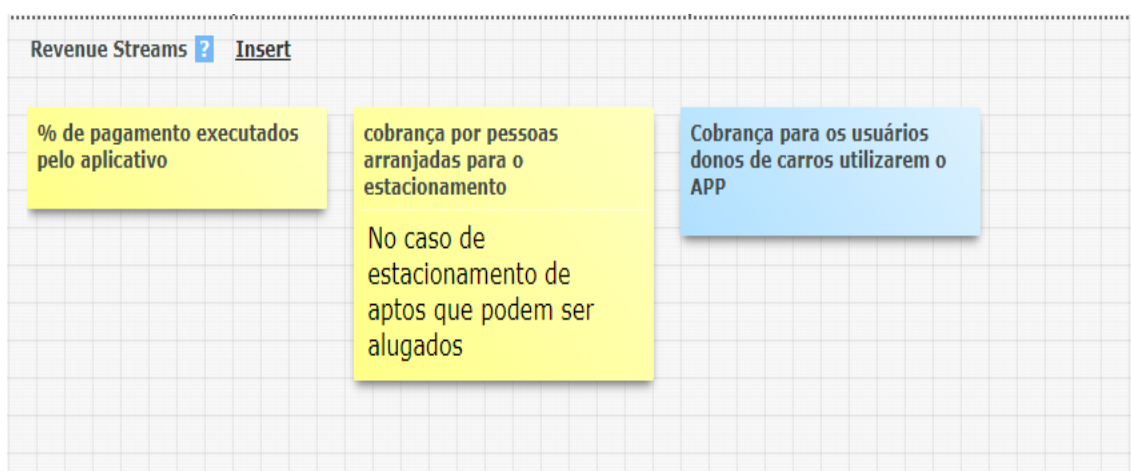
3.1.4 Quanto Custa?

A última pergunta do modelo canvas, mas não necessariamente os últimos segmentos a serem preenchidos. Como já citado no tópico anterior, a parte de receita do projeto, foi o quinto segmento a ser preenchido, visto que, tendo uma noção do que gera receita no projeto, fica mais palpável decidir se

uma atividade ou um recurso é possível ou não no projeto. Tendo isto em mente, para responder esta pergunta foram preenchidos 2 quadros do modelo canvas, sendo estes as fontes de receita, e os custos inerentes.

Para começar a responder à pergunta foi preenchido o primeiramente o quadro de fontes de receitas, assim como mostrado na figura 3.8.

Figura 3.8 - Preenchimento do segmento de fontes de receita.

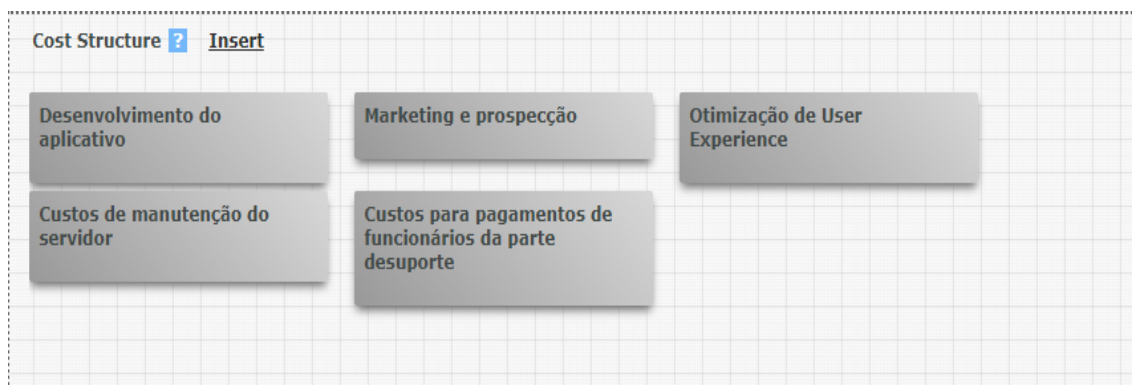


Fonte: de criação própria

Como pode-se ver na figura 3-8 existem 3 fontes de receita existentes no aplicativo, sendo elas: uma pequena quantia de pagamentos executados via aplicativo, esta receita se explica pela facilidade que será criada na forma de pagamento, tanto para o cliente, quanto para os donos de estacionamento; além desta receita foi pensado em uma cobrança por pessoa arranjada em estacionamentos de apartamentos, visto que estes não fazem muitas operações por dia, esta cobrança será bem pequena, mas que com muitos usuários pode representar uma boa parcela da receita da empresa; por fim o aplicativo que será gratuito para os usuários motorista, terá algumas propagandas, para conseguir fazer sua receita, assim como é feito em outros aplicativos de gps famosos como o Waze.

Na segunda parte para responder à pergunta "quanto custa" e o último segmento a ser preenchido do projeto, foram considerados os seguintes custos mostrados na figura 3.9.

Figura 3.9 - Preenchimento do segmento de estrutura de custos.



Fonte: Criação Própria

Assim como todos os outros segmentos a lógica dos custos estruturais do projeto, aparecem quando são visualizados a parte de atividades chave e recursos chave, algo bastante intuitivo e simples de ser completado. Da parte de custos, se fosse ser estabelecido um preço a parte de desenvolvimento do aplicativo seria a parte mais cara do projeto, visto que o aplicativo em questão não é um aplicativo simples, e também o desenvolvimento de aplicativos mobile é algo que ainda é bastante caro no mercado devido à escassez de pessoas capacitadas que trabalham nesta área, como o aplicativo está sendo desenvolvido no laboratório este custo seria zero, e só seria necessário para tornar o projeto viável uma estrutura de custos que é razoavelmente barata para começar um projeto.

3.2. Pesquisa de Mercado

Foi realizado um questionário online para ser aplicado as pessoas, atrás de entender um pouco mais sobre como as pessoas veem o problema de estacionamento em suas cidades, este formulário foi criado no google forms, e aplicado utilizando a ferramenta facebook, e pedindo para que amigos compartilhassem o formulário para que obtivesse maior número de respostas. No final da aplicação do formulário, foram obtidas

108 respostas, que possibilitou enxergar algumas coisas importantes para o mercado de algumas cidades brasileiras.

3.2.1. As Questões,

O formulário foi dividido em 5 seções, a primeira seção é uma seção para coletar dados gerais, sobre idade, sexo e se a pessoa possui carro (foi assumido que os usuários do aplicativo teriam carro), caso a pessoa possuísse veículo o questionário continuaria, caso ela não possuísse, o questionário terminaria, a figura 3.10 mostra a primeira seção do questionário.

Figura 3.10 - Primeira seção do questionário

Pesquisa Trabalho de conclusão de curso

Estou fazendo uma pesquisa para aquisição de dados para a criação de um aplicativo mobile, que será apresentado como meu trabalho de conclusão de curso.

Qual o seu Sexo? *

☐ Masculino

☐ Feminino

☐ Não quero me identificar

Quantos anos você tem? *

Texto de resposta curta

Qual cidade você mora? *

Texto de resposta curta

Você possui Carro, Moto ou Caminhão? *

☐ Sim

☐ Não

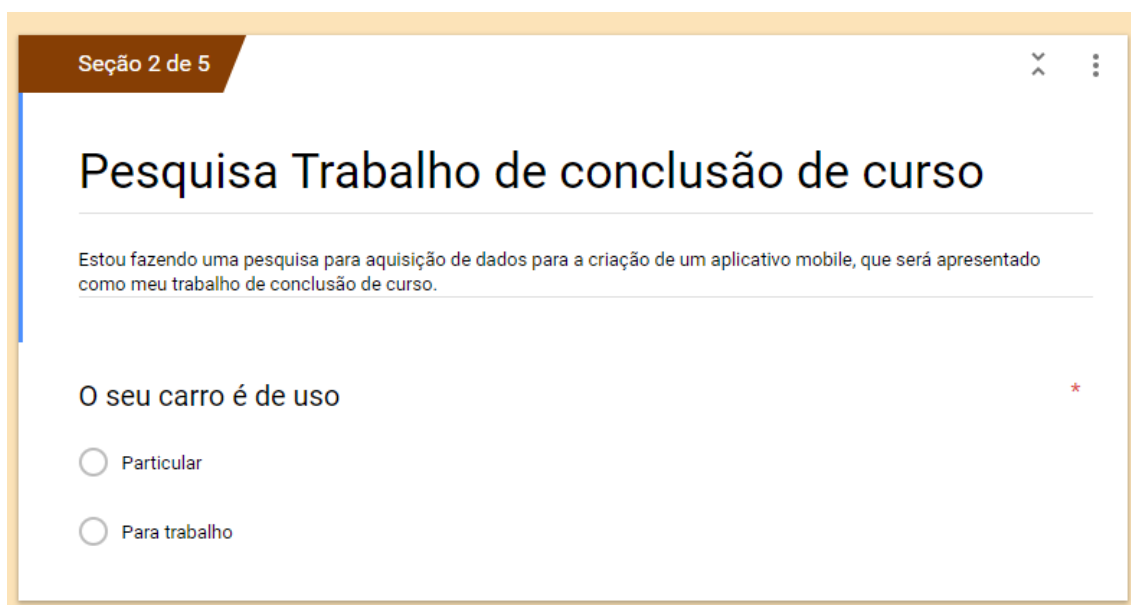
Fonte: De criação própria.

Estas informações colhidas na primeira seção do questionário, visam entender melhor qual seria o público alvo da empresa, para que tenha-se uma ideia de como a empresa iria se comunicar com o seu cliente dando indícios para uma das partes mais importantes da empresa neste primeiro momento que é a formação da marca e o tipo de

marketing que será criado. Além disso, é interessante verificar a cidade da pessoa para que fique mais fácil de entender em quais cidades seria mais interessante que a empresa começasse a sua atuação.

A segunda parte do questionário foi criada apenas para saber se a pessoa utiliza o veículo dela para trabalho, ou seja, ela trabalha como motorista de uma empresa ou entregador; ou para utilização particular, e a partir da resposta ela seria encaminhada para a seção 3 ou 4 do questionário, para responder as perguntas, de forma que as perguntas pudessem ser um pouco mais direcionadas para cada pessoa. A figura 3.11 mostra a segunda seção do questionário.

Figura 3.11 - segunda seção do questionário



Fonte: criação própria

A seção 3 e 4 são bastante semelhantes, porém com algumas perguntas diferente mais focadas na utilização dos veículos, sendo que na seção 3 a pessoa respondeu que utiliza o carro para fins particulares e a 4 ela respondeu que utiliza o veículo para fins de trabalho. As perguntas seção 3 foram criadas com a intenção de verificar se encontrar vagas é algo difícil, e atrapalha a vida das pessoas, além de verificar se estas procuram por tipos de vagas específicas, como vagas de idosos, de deficientes, gestantes etc. utilizam meios de transporte público ou se preferem ir de carro para os seus

compromissos, a frequência com a qual os indivíduos utilizam aplicativos de GPS no celular, e os tipos de vagas que elas procuram.

A seção 4 é bastante parecida com a sessão 3, portanto as perguntas tem o mesmo viés verificar a importância de encontrar vagas próximas ao local do destino, e se estes utilizam serviços de GPS em seus celulares, além disso no final das duas seções foi feita uma pergunta se a pessoa teria interesse em um aplicativo que a ajudasse a encontrar vagas, caso a resposta seja afirmativa ou em dúvida, a pessoa é levada a 5 parte do questionário, caso negativa o questionário acaba. As figuras 3.12 e 3.13 mostram as respectivas seções 3 e 4.

Figura 3.12 - Seção 3 do formulário

Seção 3 de 5

Problemas de estacionamento para pessoas com carros particulares.

Esta parte do questionário é para pessoas que possuem carro, e podem ou não ter problemas de estacionar.

É muito difícil encontrar vagas para estacionar na minha cidade. *

☐ Concordo Plenamente.

☐ Concordo.

☐ Não concordo nem discordo.

☐ Discordo.

☐ Discordo Plenamente.

Normalmente, procuro por vagas em locais:

☐ Públicos

☐ Privados (estacionamentos)

☐ Públicos e privados (tanto faz)

Fonte: Criação própria.

Figura 3.13 - Seção 4 do formulário

Seção 4 de 5

Problemas de estacionamento para carros a trabalho.

Esta parte do questionário é para pessoas que possuem carro, e podem ou não ter problemas de estacionar.

Para qual tipo de trabalho você utiliza o seu transporte? *

- ☐ Transporte de pessoas (Taxi, Mototaxi, Uber Cabify Etc)
- ☐ Transporte de mercadorias.
- ☐ Outros...

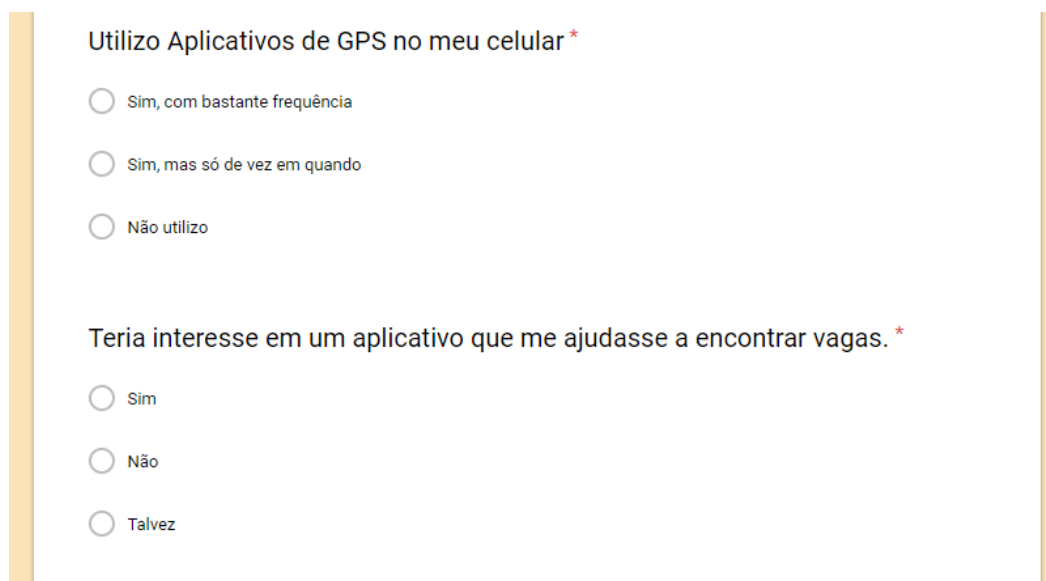
É muito difícil encontrar vagas para estacionar na minha cidade. *

- ☐ Concordo Plenamente.
- ☐ Concordo.
- ☐ Não concordo nem discordo.
- ☐ Discordo.
- ☐ Discordo Plenamente.

Fonte: Criação própria

A figura 3.14. mostra as últimas 2 perguntas colocada nas duas seções.

Figura 3.14 - Últimas questões das seções 3 e 4



Utilizo Aplicativos de GPS no meu celular *

☐ Sim, com bastante frequência

☐ Sim, mas só de vez em quando

☐ Não utilizo

Teria interesse em um aplicativo que me ajudasse a encontrar vagas. *

☐ Sim

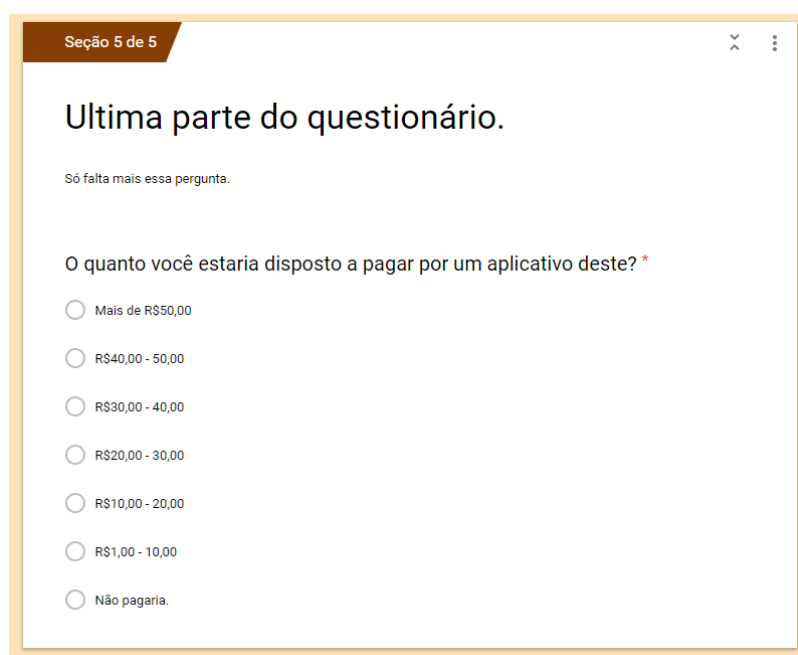
☐ Não

☐ Talvez

Fonte: Criação própria

A seção 5 é para entender se as pessoas estão dispostas a pagar pelo aplicativo, ou se a monetização dele terá de ser feita de outra maneira, como por exemplo a utilização de propagandas no aplicativo de forma a não atrapalhar a utilização dele, como é feito no aplicativo Waze por exemplo, além disso a resposta desta pergunta dará suporte aquilo que foi decidido no canvas na seção de receita (figura 3.8) e deixará claro se é necessário fazer alguma mudança. A figura 3.15 mostra a seção 5 do formulário.

Figura 3.15 - seção 5 do formulário



Seção 5 de 5

Ultima parte do questionário.

Só falta mais essa pergunta.

O quanto você estaria disposto a pagar por um aplicativo deste? *

☐ Mais de R\$50,00

☐ R\$40,00 - 50,00

☐ R\$30,00 - 40,00

☐ R\$20,00 - 30,00

☐ R\$10,00 - 20,00

☐ R\$1,00 - 10,00

☐ Não pagaria.

Fonte: Criação própria

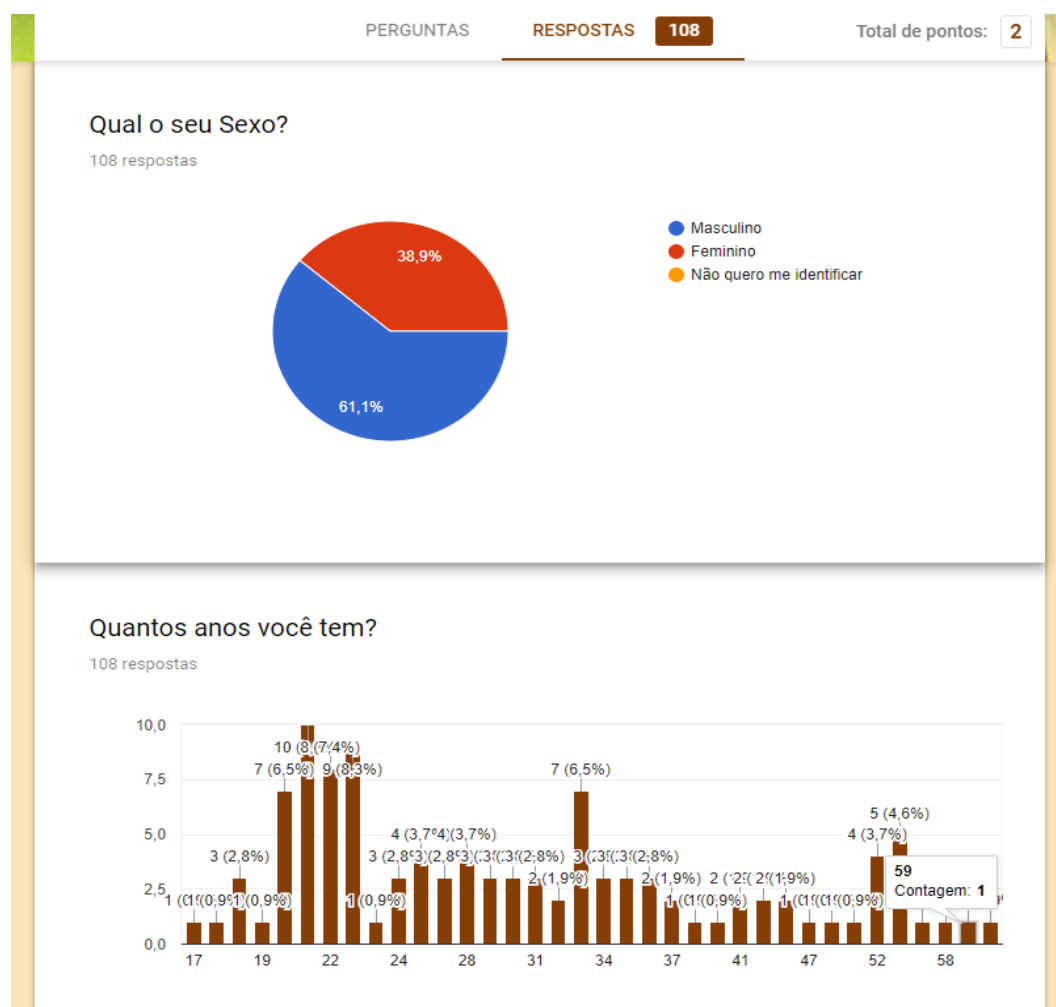
3.2.2. Resultados Obtidos.

Após a aplicação do questionário foram obtidas 108 respostas, que se analisadas de forma macro, pelas seções, foram obtidos os seguintes resultados.

Seção 1:

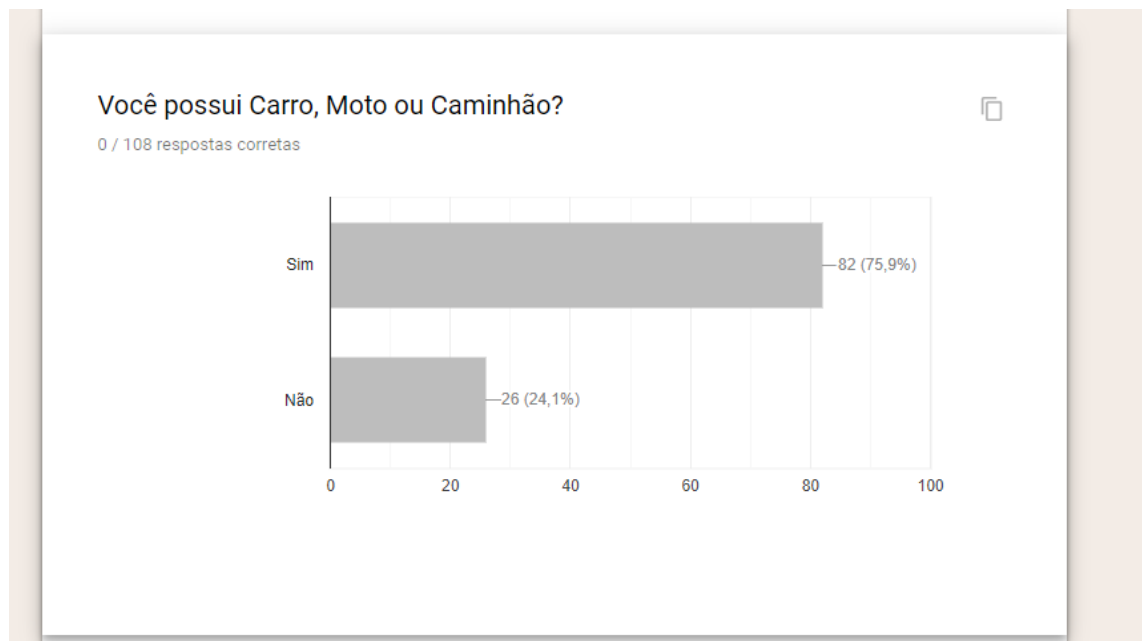
Na primeira seção tivemos uma maioria de respostas de pessoas do sexo masculino, com uma variedade bastante grande entre idades e cidades, sendo que 82% das pessoas que responderam tem carro e responderam todo o questionário, as figuras 3.16 e 3.17 mostra alguns dos resultados obtidos nas respostas da seção 1.

Figura 3.16 - Algumas respostas da seção 1 do formulário



Fonte: Criação própria

Figura 3.17 - Resultado de quantas pessoas responderam o formulário tem carros ou não



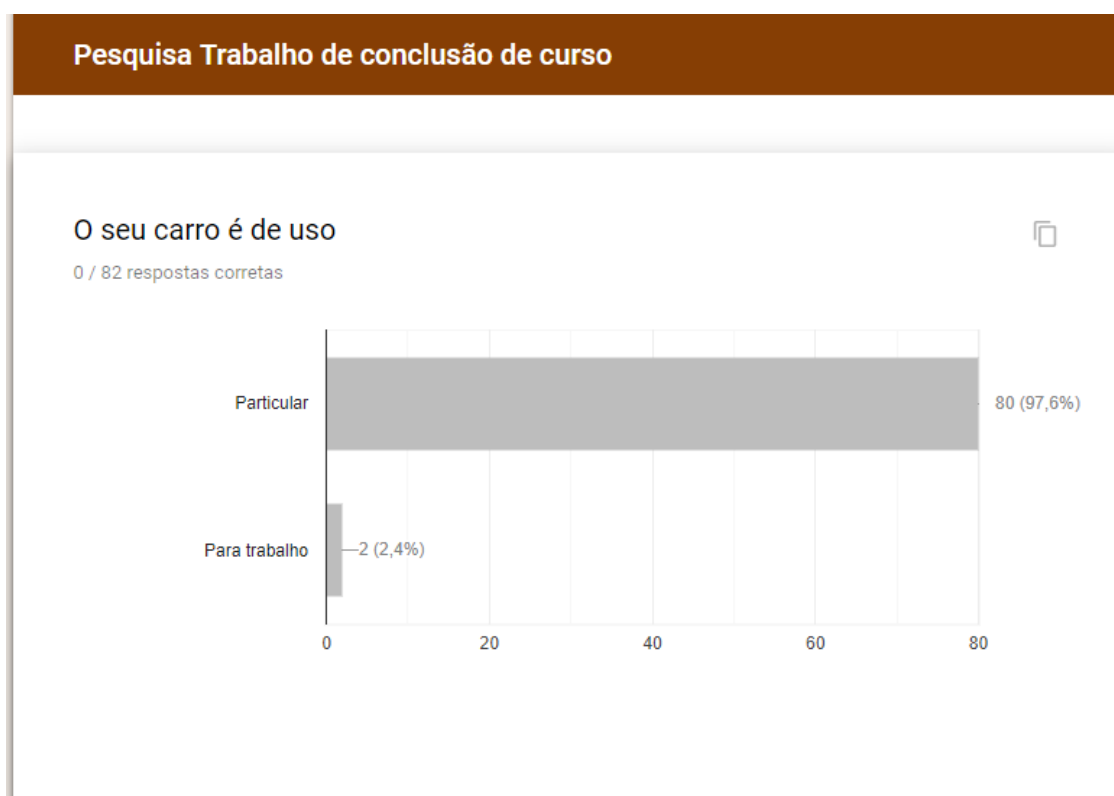
Fonte: Criação própria

Portanto é interessante segundo estas repostas focar o plano de marketing e comunicação mais voltados para o público masculino além de focar as atividades para um público mais jovem.

Seção 2:

Nesta seção, os resultados obtidos nos mostram que a maioria das pessoas que respondeu o questionário utiliza os seus veículos para uso particular. O que deixa o teste um pouco desbalanceado para saber se o aplicativo seria interessante para todas as pessoas, então o foco deste aplicativo a partir de agora será mais para as pessoas que utilizam o carro para uso particular. A figura 3.18 mostra os resultados obtidos na seção 2.

Figura 3.18 - Respostas obtidas na segunda seção do questionário



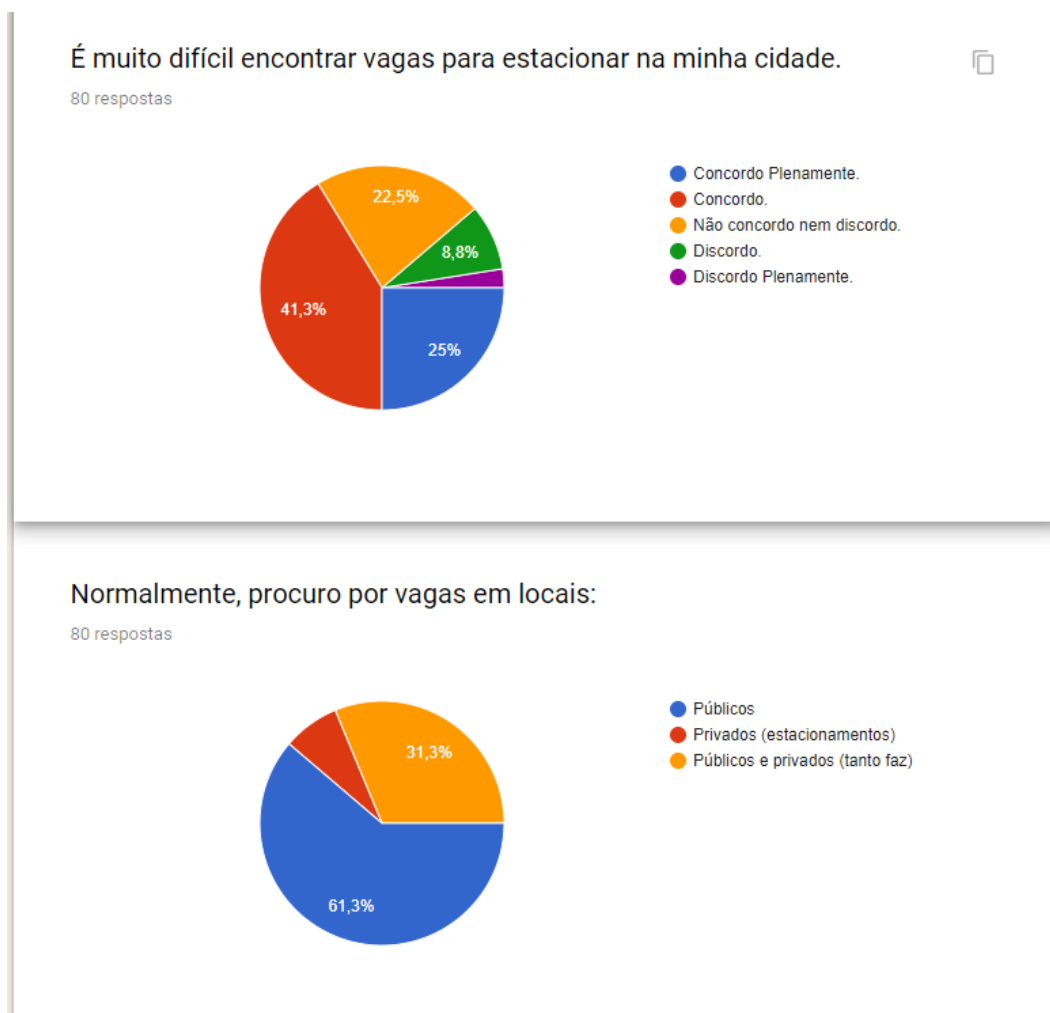
Fonte: Criação própria

Seção 3:

Nesta seção existem muitos resultados a serem comentados, primeiramente, pessoas tendem a concordar que é difícil encontrar vagas nas cidades dela, ou seja, isso realmente é um problema para a maioria das pessoas, para ser exato, 25% das 82 pessoas concorda plenamente, 41,3% concorda e 22,5% não concorda e nem discorda, logo, 66,3% das pessoas que responderam este questionário (quase dois terços) vê a dificuldade de estacionar em suas cidades como um problema, e apenas 11,2% não entendem a falta de vaga como um problema, visto que 22,5% não tem absoluta certeza se este empecilho é relevante e afeta as suas vidas.

Continuando com as perguntas desta seção, fica claro que a maioria das pessoas procuram por vagas em lugares públicos, e pouquíssimas pessoas procuram lugares privados como prioridade, o que já era esperado, visto que locais privados tendem a cobrar para que pessoas estacionem naquele local, e é comum que as pessoas queiram economizar dinheiro quando se trata de estacionar os seus carros. A figura 3.19 mostra os resultados obtidos para estes dois casos que acabamos de citar.

Figura 3.19- Resultados obtidos sobre dificuldade de estacionamento, e local procurado para estacionar



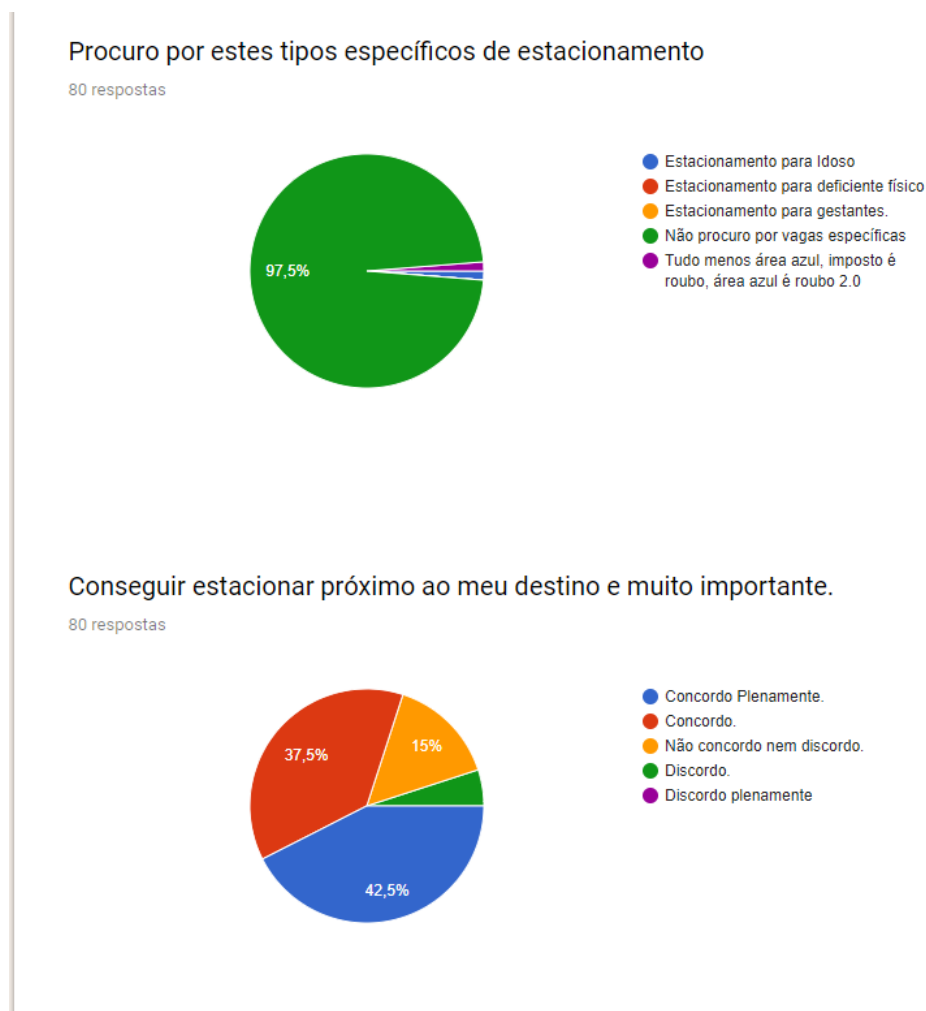
Fonte: criação própria

Continuando a olhar os resultados, as pessoas tendem a não procurar um lugar específico quando estão procurando um lugar para estacionar, para ser mais específico, nesta parte do questionário 97,5% das pessoas não procuram um lugar específico quando estão procurando um lugar para estacionar.

Além disso para a maioria das pessoas é importante estacionarem próximas ao seu local de destino, sendo que 42,5% concordam plenamente, 37,5% concordam e 15% não concorda e nem discorda. Ou seja, ter de estacionar longe dos seus locais de destino é considerado algo ruim para 80% do grupo que respondeu o questionário, portanto, ter algo que eles possam utilizar para facilitar a encontrar vagas mais próximas de onde os

indivíduos irão estacionar é algo interessante. A figura 3.20. mostra os resultados citados neste e no parágrafo acima.

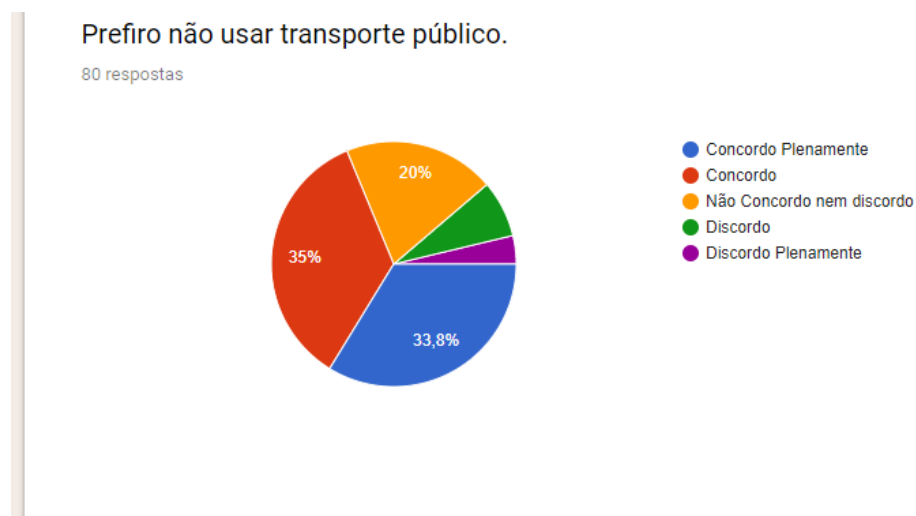
Figura 3.20 - Resultados obtidos sobre local de estacionamento, e o quanto é importante estacionar próximo ao local de destino



Fonte: criação própria.

A pesquisa também mostra que normalmente as pessoas tendem a evitar utilizar o transporte público, o que também favorece a utilização do aplicativo, visto que as pessoas necessitam de vagas para os seus carros quando vão sair de casa. Para ser mais preciso com os números, 68,8% (mais de 2/3) das pessoas preferem não utilizar transporte público, e 20% (1/5) não tem certeza se isso é um problema para elas. A figura 3.21 mostra os resultados obtidos para esta parte do questionário.

Figura 3.21 - Resultados obtidos das pessoas que preferem ou não utilizar transporte público



Fonte: criação própria

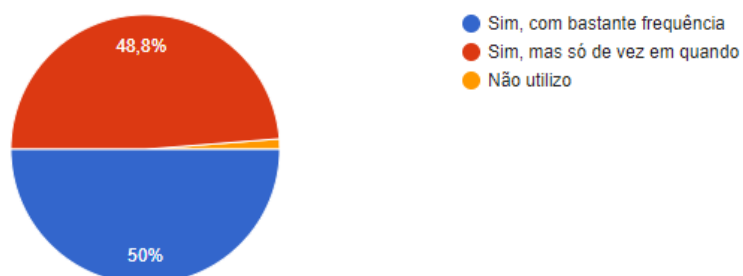
Além disso, as pessoas que responderam o questionário em sua maioria, tendem a utilizar o gps do celular quando estão dirigindo, sendo que 50% das pessoas responderam que usam o gps e 48,5% marcaram que utilizam o gps, mas com baixa frequência, ou seja, some 1,5% dos indivíduos que responderam este questionário não utilizam o gps do celular quando estão dirigindo, o que é algo bastante interessante para a utilização do aplicativo.

Por fim, a maioria dos cidadão que respondeu o questionário (85%) se interessou pela ideia de um aplicativo que os ajudaria a encontrar vagas, e apenas 3,7% acharam a ideia totalmente desinteressante, o que faz com que a chance de o aplicativo que será criado, seja de fato útil para as pessoas. A imagem 3.22 mostra os resultados comentados acima.

Figura 3.22 - Resultados obtidos para entender se as pessoas utilizam o gps do celular, e se seriam adeptas a um aplicativo para ajuda-las a encontrar vagas.

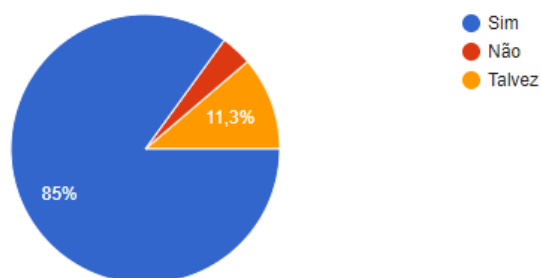
Utilizo Aplicativos de GPS no meu celular

80 respostas



Teria interesse em um aplicativo que me ajudasse a encontrar vagas.

80 respostas



Fonte: Criação própria.

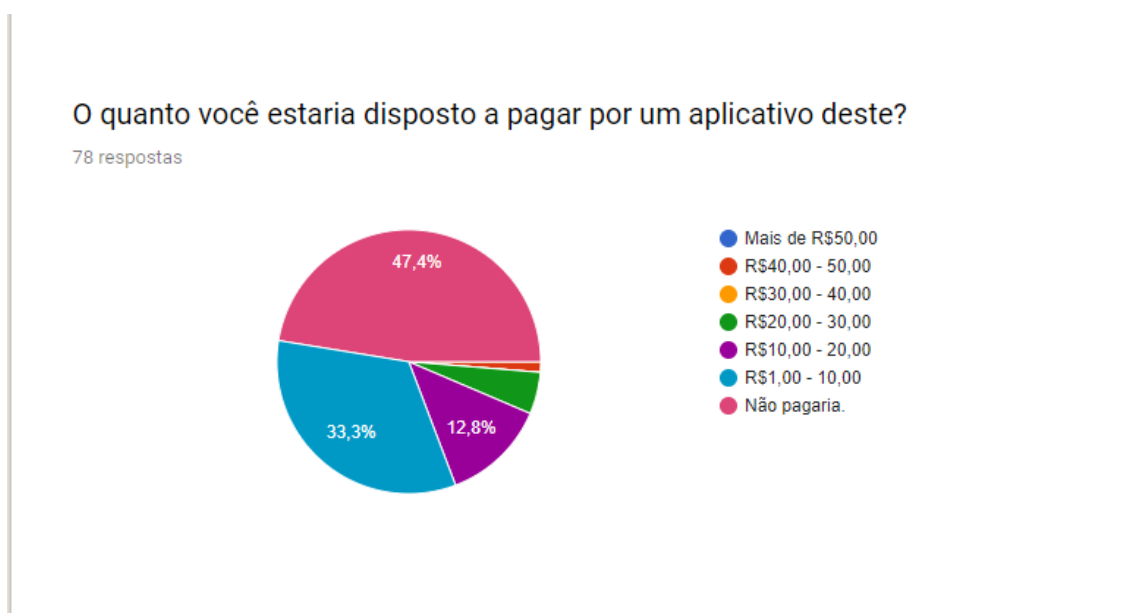
Seção 4:

Como foi obtido um número muito pequeno de respostas, esta seção será ignorada, já que, as respostas obtidas não representam fielmente algo que possa ser esperado da média das pessoas que utilizam os seus carros como ferramentas de trabalho.

Seção 5:

Nesta seção foi feita uma única pergunta para entender se o aplicativo pode ser monetizado, e também para entender melhor quanto as pessoas estariam dispostas a pagar por um aplicativo deste, a partir dos resultados, é possível entender, que apesar da falta de vagas ser um problema para a maioria das pessoas, a maior parte das pessoas não estaria disposta a pagar por um aplicativo destes (47,4%), entretanto, uma parcela grande pagaria por este aplicativo se o preço dele fosse baixo pequeno, até 10 reais (33,3%) e ainda uma parcela pequena, mas relevante para fins de estudo pagaria por um preço também não muito grande de 10 a 20 reais (12,8%). A Figura 3.23 mostra os resultados citados acima.

Figura 3.23 - Resultados obtidos para entender se as pessoas pagariam pelo aplicativo

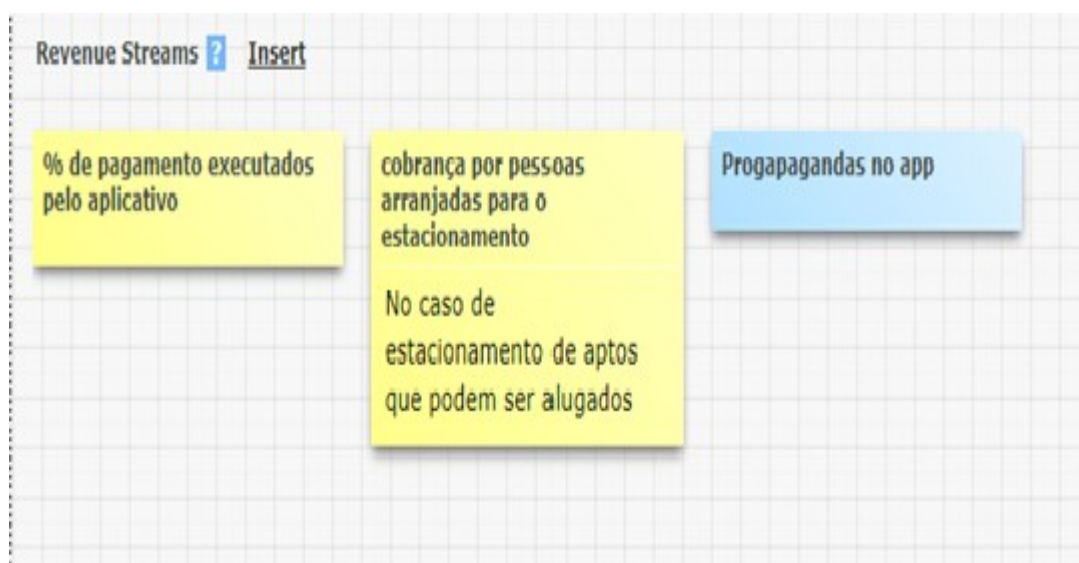


Fonte: criação própria

Através deste gráfico é possível verificar que cobrar pelo aplicativo, não seria algo muito interessante, como foi proposto no modelo Canvas (figura 3.8), portanto a opção para conseguir mais usuários pagantes, seria monetizar o aplicativo somente através de transações feitas pelo aplicativo, o que não parece ser algo muito viável, visto que a maioria das pessoas procuram vagas em lugares públicos, das quais não existem

transações de dinheiro a ser feita, e/ou então monetizar o aplicativo através de propagandas. Pesquisando um pouco outros aplicativos grandes de GPS, como o Waze, verifica-se que é possível monetizar o aplicativo com propagandas, então tendo as propagandas aliadas as quantias cobradas das transações dos clientes para locais privados é possível que o aplicativo seja monetizado de forma eficiente, entretanto foi preciso fazer alterações na parte de receitas do Canvas, a figura 3.24 mostra como essa parte ficou após a alteração.

Figura 3.24 - Seção de Receitas com as mudanças vistas necessárias pela pesquisa de mercado.



Fonte de criação própria.

3.3. Definição das atividades e desenvolvimento da matriz GUT.

Neste tópico serão definidas as atividades desenvolvidas durante a programação do aplicativo, tratando os recursos necessário que o aplicativo deve ter para funcionar, e após isso fazer através da ferramenta da qualidade matriz GUT verificar qual a melhor ordem para a execução das tarefas.

3.3.1. Definição das atividades.

As atividades para o desenvolvimento do aplicativo estão muito relacionadas com as partes e funcionalidades do aplicativo, portanto a maioria destas são tarefas de estudo da linguagem ou das APIs fornecidas pelo Corona ou de desenvolvimento (programação) do software do aplicativo, e estas serão listadas na forma de tópicos, com algumas explicações rápidas do que se trata a atividade.

- Criação e ajustes do programa para a plataforma e para os celulares: Esta atividade basicamente é a programação de 2 partes bastantes pequenas do aplicativo, porém essenciais para o bom funcionamento do mesmo na plataforma para a qual ele está sendo desenvolvido, no caso para a plataforma Android. Os arquivos que surgem desta parte do aplicativo são os arquivos `building.setting` e o arquivo `config.lua`, sendo que o primeiro faz pede as permissões de uso de alguns sensores e funcionalidades do celular, como uso da internet e uso do gps, que precisam ter permissão para serem utilizados no caso do Android; o segundo arquivo ele faz os ajuste do tamanho da imagem na tela do celular, além de também definir se o aplicativo vai girar quando girar a tela, ou então se ele deve ser usado em uma única orientação e etc.
- Desenvolvimento de um sistema de menu: O aplicativo que será criado, é um aplicativo que irá utilizar mais de uma tela, ou seja, o usuário deve poder mudar de uma tela para outra para conseguir utilizar de uma forma melhor as funcionalidades do aplicativo, para que isso seja feito é necessário que o usuário consiga abrir um menu de algum modo no aplicativo para selecionar a funcionalidade que ele quer ou precisa usar naquele momento.
- Estudar o sistema de cenas para a criação de várias telas: No framework utilizado para o desenvolvimento de aplicativos móveis (Corona SDK) existe uma biblioteca chamada `scene` (cenas em português) e esta biblioteca tem as funções para fazer as trocas de tela do aplicativo.
- Estudar o sistema de cadastro e banco de dados: Para que o aplicativo funcione será necessário um sistema de cadastro, para que os usuários possam ter as suas próprias contas, conseguindo colocar vagas ou efetuar pagamentos através dos

seus usuários. Para que isso seja possível é necessário um sistema de banco de dados para guardar estas informações sem que elas se percam.

- Desenvolvimento de um sistema de login: É necessário que a pessoa seja capaz de fazer o seu login para que ela consiga ter as funcionalidades dentro do aplicativo com o seu usuário.
- Criação de um local de cadastro de cartão: O usuário deve poder fazer o cadastro dos seus cartões de crédito, para que ele consiga executar a reserva paga de uma vaga diretamente pelo aplicativo, por conta disso é necessário que exista uma área dentro do aplicativo para que ele possa registrar os seus cartões e estes fiquem salvo dentro de um banco de dados.
- Estudar a utilização do GPS com a programação no Corona: Para que este aplicativo consiga funcionar adequadamente, e para que seja possível visualizar as vagas disponíveis em um mapa na tela do celular, será necessário estudar como o Corona utiliza o GPS e as funções disponíveis para este sensor do celular.
- Desenvolvimento de um cadastro de vagas: Nesta parte do programa é necessário que o usuário consiga cadastrar um estacionamento e as vagas que ele irá oferecer para os outros usuários, estes dados assim como os anteriores serão guardados dentro de um banco de dados para que estes dados sejam guardados e acessados pelos aplicativo para fazer as marcações no mapa.
- Fazer a portabilidade dos bancos de dados para online para acesso remoto: É necessário que os bancos de dados estejam na nuvem para que o aplicativo consiga juntar a informações dadas por vários usuários diferentes e faça com que o aplicativo consiga disponibilizar as informações de vagas existentes de todos os usuários e marcar no mapa GPS onde estas estão.
- Implementação de um mapa no menu do programa, fazendo as marcações no mapa das garagens registradas: O usuário ao entrar no programa deve conseguir ver o mapa GPS para que ele consiga saber aonde estão disponibilizadas as garagens registradas no aplicativo, para isso é necessário criar um objeto de mapa na programação lua, utilizando as APIs disponibilizadas pelo Corona.
- Criar um sistema de reserva de garagem: Esta parte do aplicativo faz com que o usuário possa reservar uma vaga em uma determinada garagem por um determinado tempo em um determinado dia, e no horário que esta vaga estiver reservada, ninguém mais possa reservar a garagem.

3.3.2. Matriz GUT.

Após executada a definição das tarefas, estas foram avaliadas dentro dos requisitos da Matriz GUT (Gravidade, Urgência, Tendência), os resultados obtidos na ferramenta serão mostrados na figura 3.24. a seguir.

Figura 3.25 - Resultados obtidos na análise GUT

Atividades	Gravidade	Urgência	Tendência	Prioridade
Criação dos ajustes do programa para a plataforma	3	5	2	10
Desenvolvimento de um sistema de menu	3	5	1	9
Estudar sistema de cenas, para criação de várias telas	4	4	3	11
Estudar sistemas de cadastros e bancos de dados	4	4	3	11
Criação de um sistema de cadastro	5	4	1	10
Desenvolvimento de um sistema de login	5	4	1	10
Criação de um local de cadastro de cartões	1	3	1	5
Estudar utilização do GPS, dentro da programação	4	3	3	10
Desenvolvimento de um cadastro de endereços	5	3	1	9
Fazer a portabilidade dos bancos de dados para Online com acesso remoto	5	2	1	8
Implementação de um mapa no menu do programa, fazendo as marcações no mapa das garagens registradas	4	2		6
Criar um sistema de reserva de garagens.	3	1	1	5

Fonte: Matriz desenvolvida pelo laboratório LINA no software Excel

Como pode ser notado na matriz GUT, as atividades executadas não tem notas de tendência muito elevadas, isso se deve pelo fato, destas atividades, não ficarem mais difíceis ao longo do tempo, as atividades que tem uma nota de tendência são atividades de estudo normalmente, e isso se deve ao fato de que quanto mais tempo o aluno demora para fazer as atividades de estudo mais tempo ele demora para desenvolver as outras atividades, além de que implementar algo sem o estudo prévio pode fazer com que a programação fique errada e tenha de ser refeita, ou seja quanto mais tempo demora-se para estudar os assuntos citados, mais difícil fica de fazer a implementação do programa

no tempo desejado. Tendo este fato explicado da nota de tendência, agora será explicado individualmente o porquê das notas dadas para cada atividade que será executada, em forma de tópicos para ficar um pouco mais simples de entender.

- Criação e ajustes do programa para a plataforma e para os celulares: Esta parte do programa é bastante importante que ela seja programada primeiro, para que seja mais fácil de fazer as adaptações na tela, esta parte é bastante crítica pois é nela que são pedidas as permissões do aplicativo, ou seja, as funcionalidades necessárias para o funcionamento do aplicativo como o GPS e a internet, não funcionarão, logo é impossível do aplicativo rodar sem esta parte feita. Por fim, esta parte do aplicativo tem uma certa tendência devido ao fato de que, quando são feitas atualizações no sistema da Android ou IOS o jeito de programar esta parte do programa pode mudar, e as vezes essa programação demora algum tempo para ser liberada, o que faz com que no longo prazo caso haja mudanças pode acontecer de haver um aumento da dificuldade de programar o building.settings do programa, portanto a tendência desta parte foi colocada como 2.
- Desenvolvimento de um sistema de menu: O desenvolvimento do sistema de menu é uma parte bastante urgente do programa, pois programando esta parte fica bem mais simples de entender como programar as partes posteriores, visto que é o menu que ligará todas as partes, e é necessário para fazer os testes, por conta disso ele é tão urgente. Como ele é a parte que liga todos as outras telas do aplicativo de uma forma simples, isto faz com que esta parte do aplicativo seja um Crítica
- Estudar o sistema de cenas para a criação de várias telas: Como já dito antes as partes do aplicativo que requerem estudo tem já tem um valor de tendência atribuído. Como esta parte é necessária para entender todo o sistema de telas do programa, isso faz com que ela seja muito urgente e bastante crítica, visto que o aplicativo precisa de diferentes telas para conseguir fazer as diferentes funcionalidades de forma adequada.
- Estudar o sistema de cadastro e banco de dados: O sistema de cadastro e banco de dados, é o que torna o programa do aplicativo possível, sem um banco de dados, seria impossível guardar as localizações das garagens, além dos cadastros de usuários que faz com que o programa seja possível, logo entender como

implementar o cadastro e a parte de banco de dados é muito importante e bastante urgente.

- Desenvolvimento de um sistema de login: O sistema de login é o que vai controlar a pessoa e o cartão que cada pessoa pode utilizar, além de controlar quem são os donos de cada estacionamento ou vaga. Portanto ele é fundamental para o funcionamento do aplicativo. Por conta disso esta parte do aplicativo tem uma alta nota no quesito crítico, e a urgência dele é alta, entretanto, é necessário que o sistema de menu seja feito antes desta parte, por conta disso a urgência não é tão grande.
- Criação de um local de cadastro de cartão: No aplicativo completo para a parte de venda, esta parte do aplicativo é bastante importante, visto que fica bem mais fácil do usuário não precisa ficar colocando toda as vezes o seu cartão de crédito quando ele for fazer uma reserva de uma vaga. Entretanto, como o aplicativo que será feito, será um protótipo, e nenhum sistema de pagamento por cartão será contratado para este aplicativo; esta parte do aplicativo fica sendo mais um adicional só pra deixar ele mais próximo do produto final, por conta disso as notas dadas a ele na matriz são bem baixas, ele tem um pouco de urgência, pois é mais interessante estruturar o banco de dados desta parte antes de fazer a portabilidade para o sistema online, para ter que evitar retrabalho com a programação php que será utilizada.
- Estudar a utilização do GPS com a programação no Corona: A utilização do sensor GPS do celular é extremamente importante para conseguir colocar um objeto de mapa no celular, para que o usuário consiga visualizar os locais aonde estão sendo disponibilizadas as vagas, o que é muito importante, para o user experience, além de que a utilização do GPS faz com que seja mais fácil de conseguir os endereços dos usuários, o que faz com que o aplicativo seja mais funcional, portanto ele é muito grave, e como é uma parte de estudo ele também se torna de certa forma urgente e também tem uma certa tendência.
- Desenvolvimento de um cadastro de vagas: O cadastro de vagas é uma das funcionalidades principais deste aplicativo segundo o objetivo dele, logo caso não exista um cadastro de vaga, não existe o aplicativo, então esta parte do aplicativo é fundamental para o aplicativo, entretanto ela não é extremamente urgente, pois

ela precisa ser criada após alguns outros feitos, como a parte de cadastro, login e menu, para que a parte as garagens possam pertencer a alguma pessoa.

- Fazer a portabilidade dos bancos de dados para online para acesso remoto: Assim como a funcionalidade anterior, a não existência dos bancos de dados online, faz com que seja impossível que o aplicativo exista de forma funcional, visto que é necessário que todas as pessoas sejam capaz de acessar o banco de dados de endereços para que seja possível marcar no mapa os locais aonde estão localizados as garagens e vagas disponíveis.
- Implementação de um mapa no menu do programa, fazendo as marcações no mapa das garagens registradas: A implementação de um mapa no aplicativo faz com que seja muito mais fácil para os usuários saberem aonde estão localizados as vagas, portanto, esta atividade tem é bastante crítica, entretanto ela não é tão urgente, pois esta parte pode ser implementada apenas no fim do projeto.
- Criar um sistema de reserva de garagem: O sistema de reserva de garagem, é algo muito importante para o aplicativo, entretanto ele é a última coisa a ser implementada no aplicativo.

4. Desenvolvimento do aplicativo e aplicações futuras.

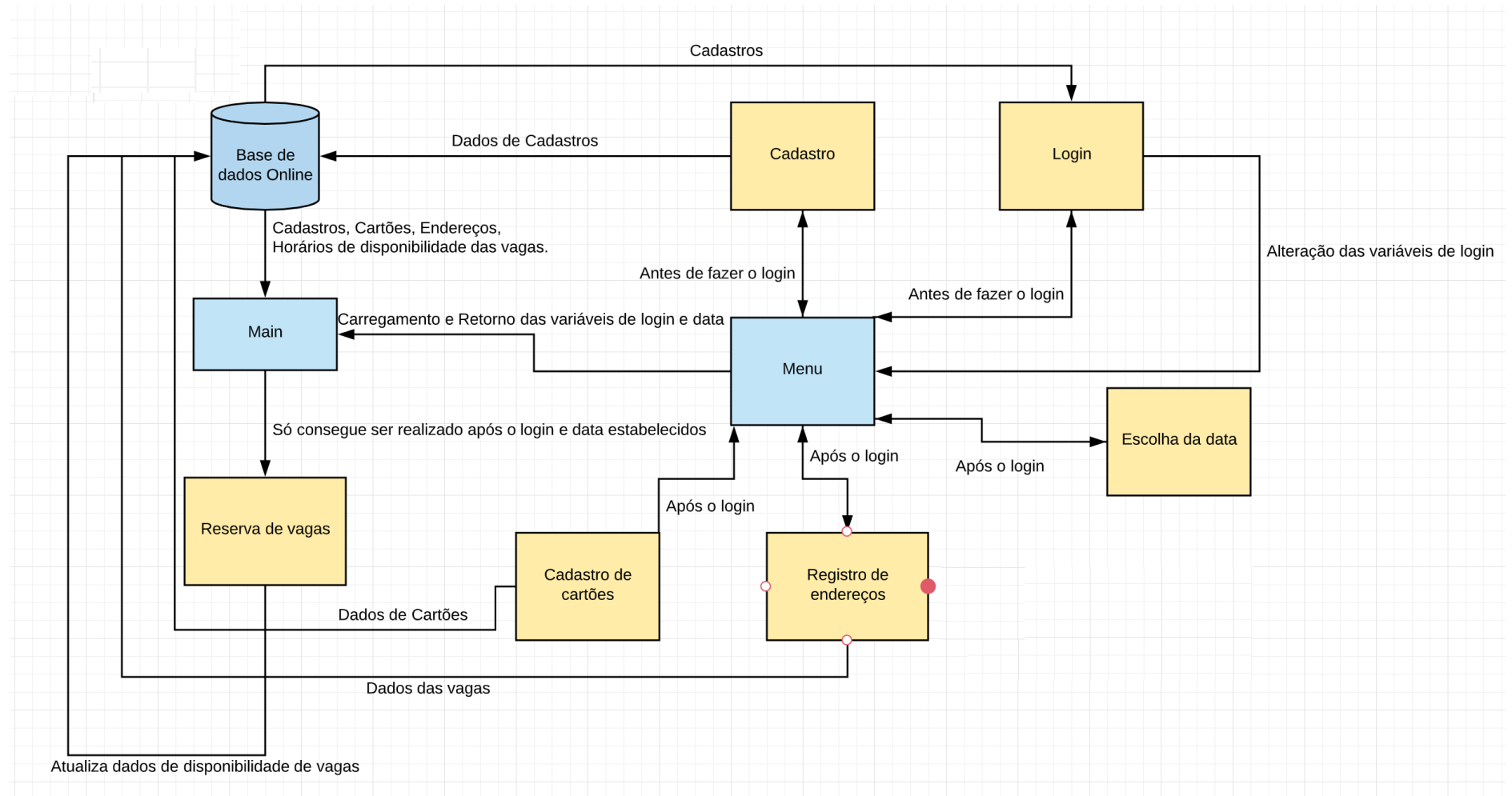
4.1. Como O App funciona.

O aplicativo foi desenvolvido pensando que ele terá uma tela principal (menu) no qual você consegue ver o mapa, além de ter acesso a uma barra de menu que dá acesso a outras janelas do aplicativo, sendo para que seja feito as opções principais do aplicativo, é necessário que o usuário tenha efetuado o login, pois é necessário do dados dessa pessoa para que ela consiga fazer a reserva. Além disso o aplicativo tem um sistema de banco de dados online, para acesso remoto, para que todos os usuários possam ter acesso aos endereços cadastrados por outros usuários, pois caso contrário seria impossível ter acesso

a estes dados; estes dados efetuados pelo banco de dados é carregado na inicialização do programa, na função main do aplicativo.

Para tentar deixar as coisas um pouco mais visual, eu criei um pequeno fluxograma para tentar exemplificar como é o funcionamento do programa, este será apresentado na figura 4.1.

Figura 4.1 - Fluxograma de funcionamento do aplicativo



Fonte: Criação original, feito no site ludichart.com.

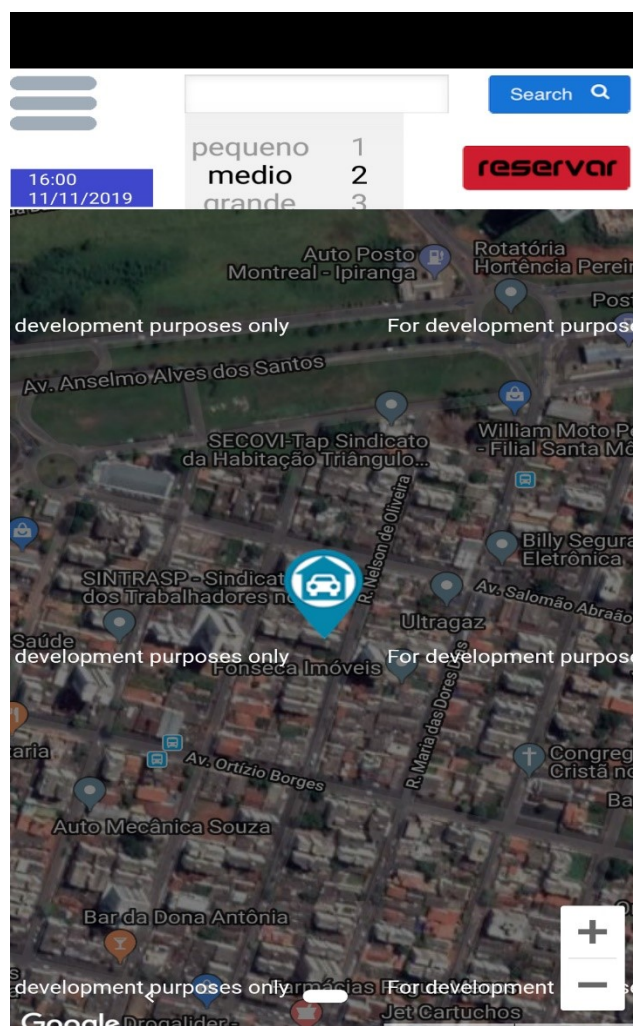
Como pode ser visto no fluxograma, quase todas as funções têm comunicação com o menu, e com exceção do menu todas as funções fazem alguma interação com o banco de dados, seja recebendo informação do banco de dados, ou enviando informações para ele. O login é uma execução bastante importante para o programa, para que os locais acessíveis pelo menu sejam modificados, os dados de login são necessários para que seja feito o controle de usuários, e quem irá fazer a reserva de vagas ou os registros de garagem (na parte de registro de endereços) e de cartões.

É importante citar, que o banco de dados criado, foi desenvolvido na nuvem, e a comunicação é feita entre o Corona (programação lua) com o servidor do HostGator (programação php), para ficar um pouco mais fácil a programação o aplicativo ao iniciar ele baixa os bancos de dados para o celular, o que faz com que menos interações entre duas linguagens diferentes sejam feitas.

Para ficar um pouco mais fácil de entender, explicarei separadamente cada parte do programa, na forma de tópicos:

- Main / Reserva de vagas: Estas partes do programa foram desenvolvidas em um único arquivo de código, por conta disso serão explicadas juntas. O main carrega as variáveis de login, que como padrão são variáveis para que o programa entenda que o usuário não tenha feito login, além disso ele recebe os dados direto banco de dados para que seja feito o carregamento do mapa com as devidas marcações de onde se encontram as garagens disponíveis cadastradas por outros usuários, A figura 4.2 mostra a parte principal do Aplicativo.

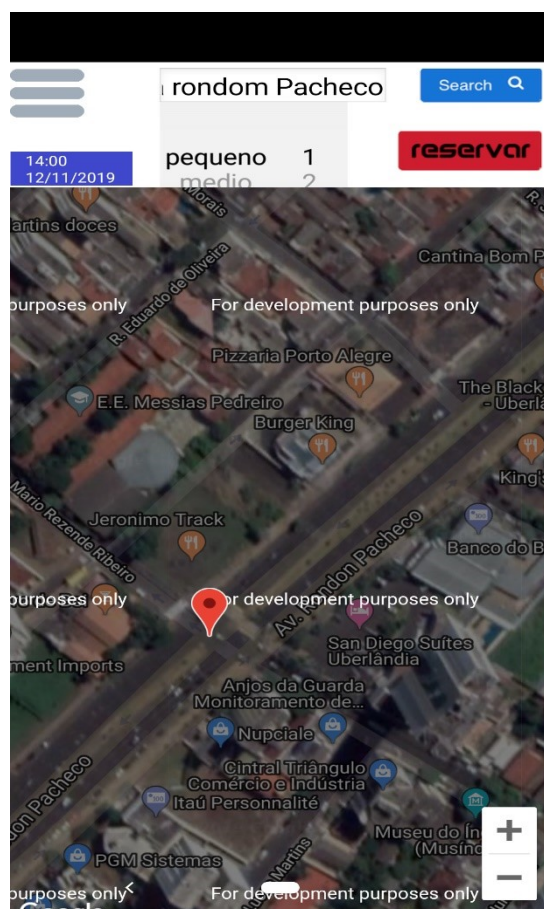
Figura 4.2 - Parte principal do aplicativo.



Fonte de criação própria

Além disso o main contém a parte de reserva de vagas, logo nele existe a barra de procura, que faz com que um endereço específico seja mostrado no mapa, assim como mostrado na figura 4.3.

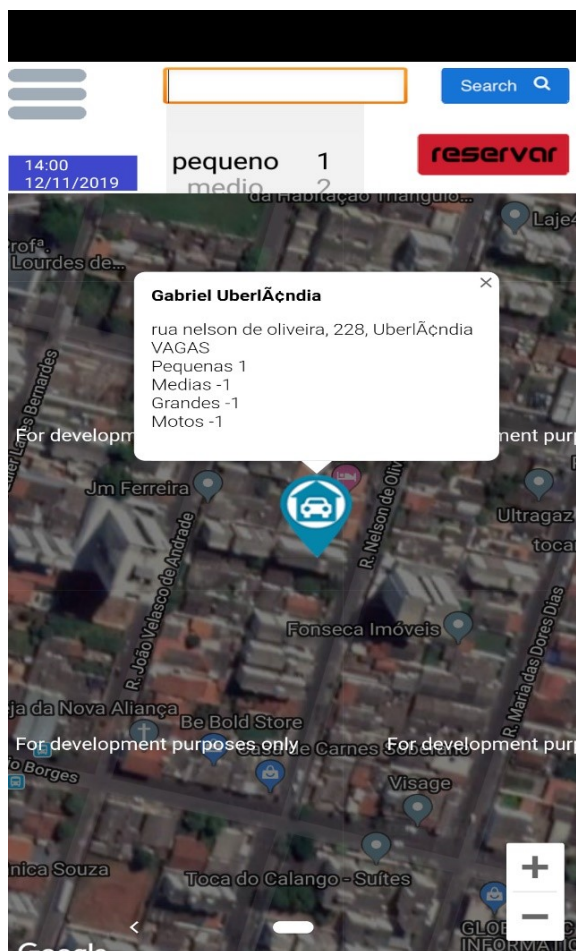
Figura 4.3 - Figura que mostra um local marcado no mapa encontrado pela barra de procura



Fonte de criação própria.

Esta parte do App também contém uma roda para selecionar o tamanho e a quantidade de vagas necessárias que serão solicitadas; além disso existe um display no canto da tela que mostra a data e hora selecionada pelo usuário (a determinação de data e hora é feita na parte data do programa). Por fim existe o botão de reserva que irá pegar os dados de uma garagem selecionada no mapa, que é representada por um carrinho azul, assim como mostrado na figura 4.4, a data, a hora a quantidade e tamanho das vagas (selecionada na roleta que se encontra no topo da tela) e registrar no banco de dados de reserva; ao fazer este processo, o aplicativo mostrará uma mensagem de confirmação com todos os dados selecionados para que o usuário não faça a escolha errada, figura 4.5; e depois ele mostra um alerta avisando que o cadastro no banco de dados foi realizado com sucesso e a vaga está reservada, figura 4.6.

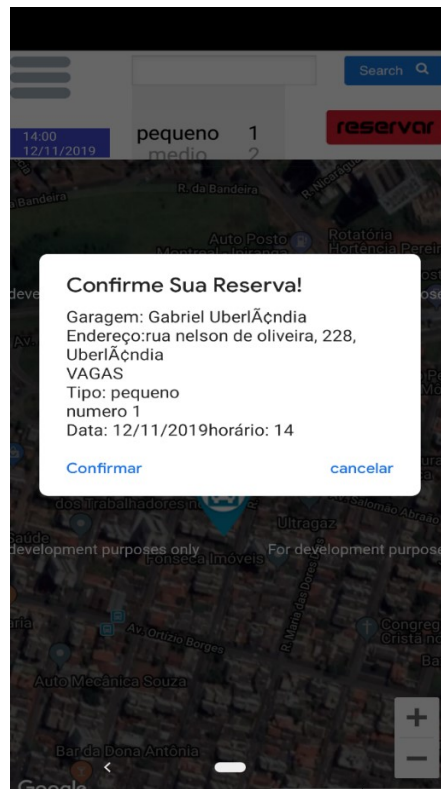
Figura 4.4 - Figura que mostra como o usuário pode identificar uma vaga no aplicativo, e as informações da vaga caso o usuário toque nela.



Fonte de criação própria

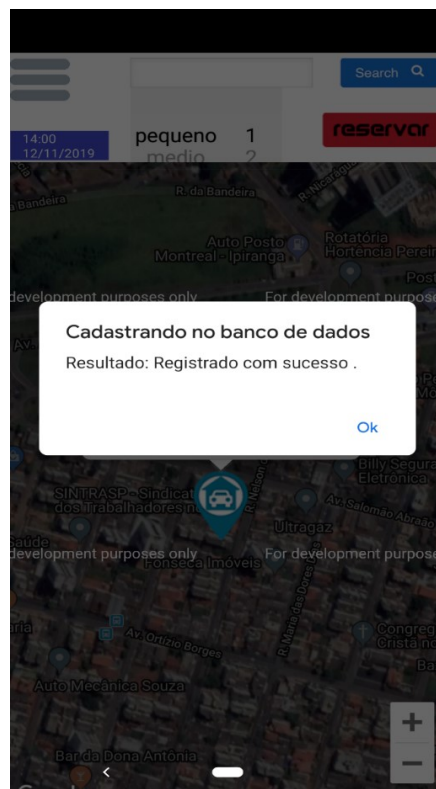
É possível verificar que ocorreu um erro no momento de mostrar o nome da cidade bastante comum na programação, que é um erro envolvendo caracteres especiais. O ‘â’ da palavra Uberlândia não foi reconhecido corretamente, e para o aplicativo completo seria uma das coisas que deveria ser observada.

Figura 4.5 - Figura que mostra a mensagem de confirmação de reserva de vaga.



Fonte de criação Própria

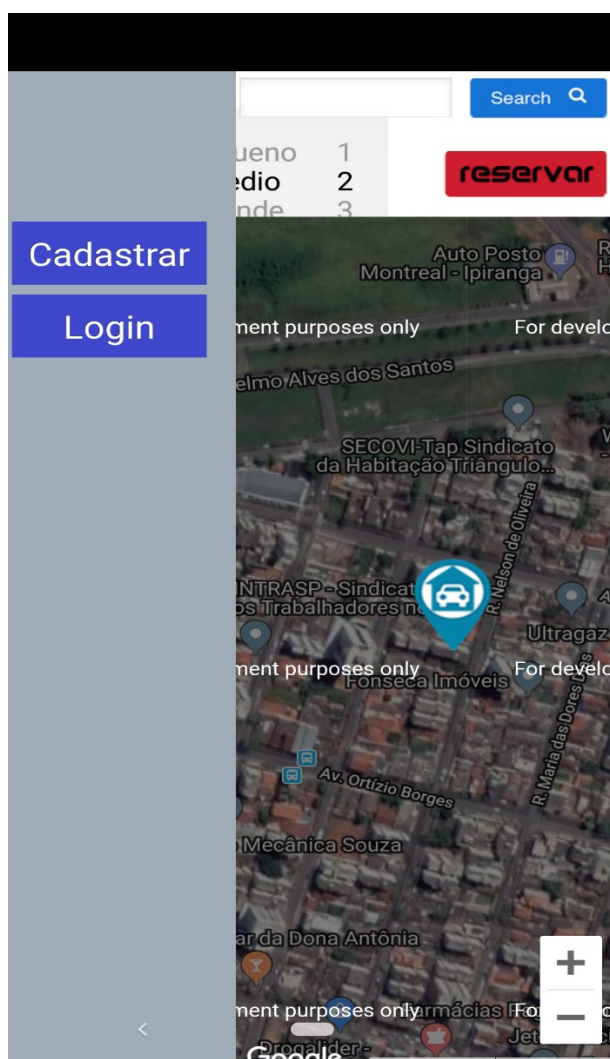
Figura 4.6 - Alerta de que a reserva foi feita com sucesso.



Fonte de criação Própria.

- Menu: Esta parte do programa, é um botão que fica no canto superior esquerdo do Main, que ao ser clicado abre pequena aba que fica do lado esquerdo e que dá acesso a todas as outras partes do programa, sendo que o conteúdo desta aba muda, antes do login, ela dá acesso apenas a área de login e de cadastro, assim como mostrado na figura 4.7;

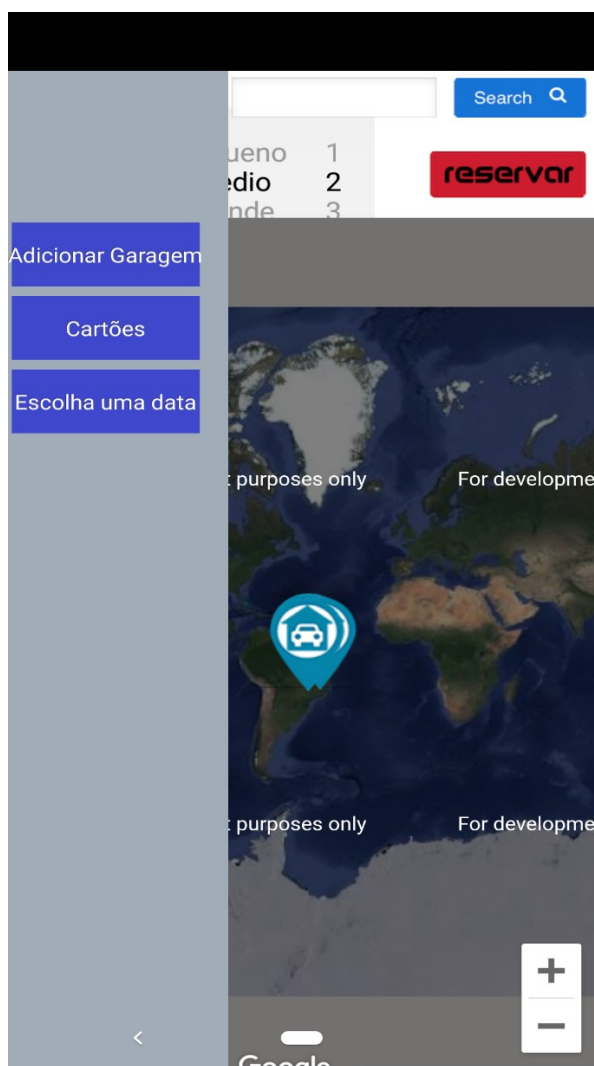
Figura 4.7 - Aba menu aberta antes que o login tenha sido feito.



Fonte de criação própria.

e após o login ela dá acesso a parte de cadastro de vagas, cadastro de cartões e a parte de selecionar a data, assim como mostra a figura 4.8.

Figura 4.8 - Aba menu aberta após a execução do login



Fonte de criação própria.

- Cadastro: Nesta parte do aplicativo o usuário deve criar o seu cadastro para que ele consiga efetuar o login no APP. Posteriormente. Nesta janela a pessoa que está efetuado o cadastro deve escrever as suas informações, manualmente, e após isso clicar na tecla de enviar para que os dados sejam passados para o banco de dados online, e depois retornar a tela de menu. As informações requisitadas no cadastro são: o nome do usuário, o e-mail, o login, a senha, confirmar a senha e o CPF dele; caso as senhas colocadas nas caixas de texto forem diferentes, o programa fará um aviso de erro, e continuará na mesma tela. A figura 4.9 mostra o resultado da tela de cadastro, e a figura 4.10 mostra o erro quando as senhas são diferentes.

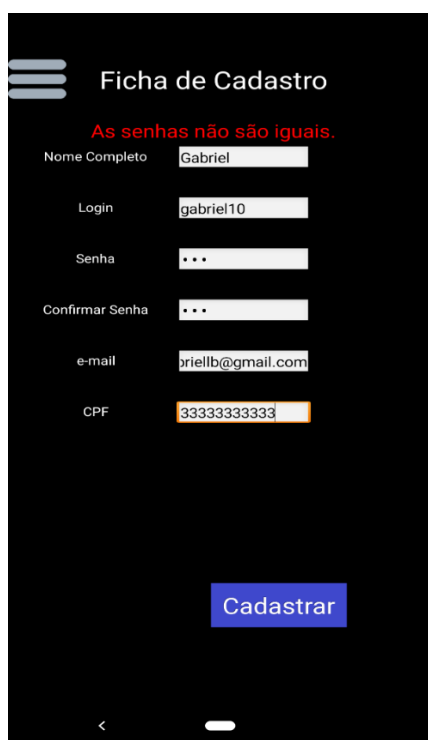
Figura 4.9 - Área de cadastro de membros



The screenshot shows a mobile application window titled 'Criando botão para o TCC - ...'. The app interface is titled 'Ficha de Cadastro' and features a dark background with white text. It includes input fields for 'Nome Completo', 'Login', 'Senha', 'Confirmar Senha', 'e-mail', and 'CPF'. A blue 'Cadastrar' button is positioned at the bottom of the form.

Fonte de criação Própria

Figura 4.10 - Mensagem avisando que as senhas são diferentes.



This screenshot shows the same 'Ficha de Cadastro' screen as Figure 4.9, but with an error message in red text: 'As senhas não são iguais.' (The passwords are not equal). The input fields are filled with the following data: 'Nome Completo' is 'Gabriel', 'Login' is 'gabriel10', 'Senha' and 'Confirmar Senha' are masked with dots, 'e-mail' is 'riellb@gmail.com', and 'CPF' is '33333333333'. The blue 'Cadastrar' button remains at the bottom.

Fonte de criação própria

Caso o Usuário coloque um CPF inválido, o programa também mostrará uma tela de erro, visto que o CPF é a chave primária do banco de dados, além de através dele conseguirmos descobrir tudo que pertence a pessoa cadastrada, como cartão e até vagas. Um CPF é considerado válido quando a soma do 1º número dele multiplicada por 10 mais o 2º número multiplicado por 9 mais o 3º número multiplicado por 8... até o 9º número multiplicado por 2 de um resultado no qual o resto da divisão deste por 11 seja igual ao 10º dígito do CPF ou o primeiro dígito verificador. Além disso a soma do 1º número multiplicado por 11 mais o 2º número multiplicado por 10... até o 10º número multiplicado por 2 de um resultado no qual o resto da divisão deste por 11 seja igual ao 11º dígito do CPF ou o segundo dígito verificador.

Vou explicar a lógica computacional para se fazer isso no programa, pois nessa parte dele eu utilizei uma biblioteca pronta que não é do próprio Corona SDK. Para verificar o CPF é necessário que ele seja recebido no formato de string, e através de um “loop” fazer com que o programa passe por todos os valores da string e salve cada valor desta em uma table ou em 11 variáveis diferentes (a table é mais otimizado). Após isso deve ser criado um vetor peso com os valores (10, 9, 8, 7, 6, 5, 4, 3, 2) e então deve ser feito o somatório do 1º valor da table multiplicado pelo 1º valor do vetor peso até o 9º valor da table multiplicado pelo 9º valor do vetor peso. Este resultado deve ser encontrado o resto da divisão dele por 11 e guardado em uma variável. Em seguida deve-se criar um vetor peso2 com os valores (11, 10, 9, 8, 7, 6, 5, 4, 3, 2) e repetir o somatório anterior só que dessa vez com os 10 primeiros valores da table e do vetor peso; e então deve-se encontrar o resto da divisão do resultado por 11 e guardar em uma outra variável. Após feito isso é necessário verificar se alguma das 2 variáveis encontradas no final de cada processo é maior do que 9, caso ela seja essa variável deve ser substituída por 0. Por fim checa-se se a variável encontrada no primeiro processo é igual ao 10º valor da table e a segunda variável encontrada no processo é igual ao 11º valor da table, se os 2 casos forem verdade, o CPF é verdadeiro.

A figura 4.11 mostra a mensagem de erro que aparece na tela caso, o usuário entre com um CPF inválido.

Figura 4.11 - Mensagem de Erro caso o usuário entre com um CPF inválido

A imagem mostra uma interface de usuário para uma tela de cadastro, intitulada "Ficha de Cadastro". No topo, há um ícone de menu hambúrguer. Abaixo dele, a mensagem de erro "O CPF é Inválido." é exibida em vermelho. O formulário contém os seguintes campos:

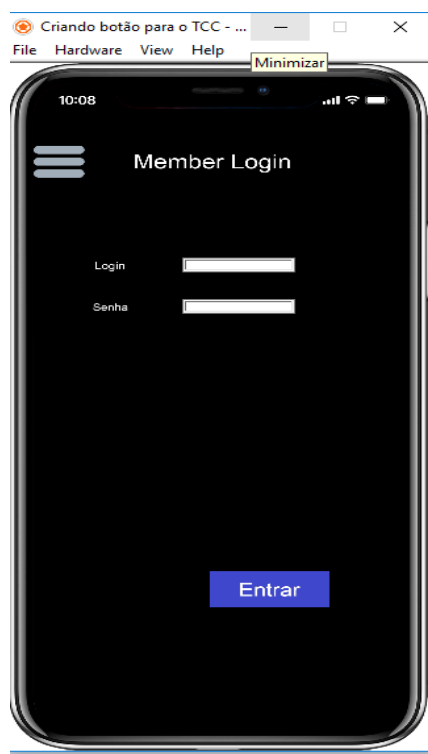
- Nome Completo: gabriel
- Login: beetazzi230
- Senha: ...
- Confirmar Senha: ...
- e-mail: abriel@gmail.com
- CPF: 44255288236

Na parte inferior do formulário, há um botão azul com o texto "Cadastrar".

Fonte: de criação própria

- Login: A área de login, assim como a tela de cadastro, é um local bastante simples, no qual o usuário vai ter 2 campos para preencher, o campo de login e o campo de senha, e após isso ele terá um botão que ao ser clicado, o programa verificará no banco de dados se o login e a senha existem, e foram cadastrados pela mesma pessoa (estão na mesma linha do banco de dados) caso isso seja verdade ele irá efetuar o login, alterando as variáveis de login e retornará para a tela de menu, caso contrário ele escreverá em vermelho no topo da tela que o login ou senha estão incorretos, e continuará na mesma tela. A figura 4.12. mostra como a tela de login, e a figura 4.13 mostra a mensagem de erro.

Figura 4.12 - Tela de login



Fonte: de criação própria

Figura 4.13 - Mensagem de erro na tela de login



Fonte de criação própria

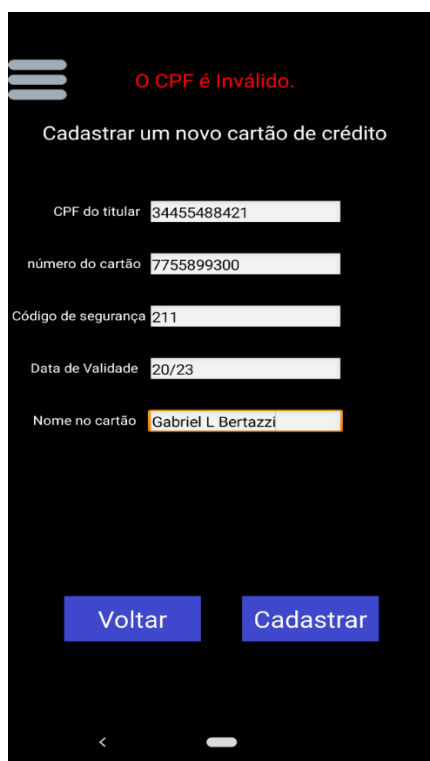
- Cadastro de cartões: A tela de cadastro de cartões de crédito também é muito similar as telas anteriores, pois assim como as outras esta tem apenas campos para escrever e um botão para efetuar o cadastro e outro para voltar para a tela anterior. Nesta parte do programa, o usuário deve entrar com o CPF do titular do cartão, o número do cartão, o código de segurança do cartão e o nome do titular do cartão. Colocados os dados, o código irá verificar se o número de CPF é válido, caso ele não seja, será criado uma mensagem de erro na tela, avisando que o CPF não pode ser utilizado, a verificação de CPF é idêntica à feita na parte de cadastro. A figura 4.14. mostra como ficará a parte de cadastro de cartões, e a 4.15 mostra o erro caso o CPF seja inválido.

Figura 4.14 - Tela de cadastro de cartões

A imagem mostra a interface de uma aplicação móvel para o cadastro de um novo cartão de crédito. No topo, há um ícone de menu hambúrguer. Abaixo dele, o título "Cadastrar um novo cartão de crédito" está centralizado. O formulário contém cinco campos de entrada, cada um com um rótulo à esquerda: "CPF do titular", "número do cartão", "Código de segurança", "Data de Validade" e "Nome no cartão". Na base do formulário, há dois botões azuis: "Cadastrar" à esquerda e "Voltar" à direita. Na barra de navegação inferior, há um ícone de seta para trás à esquerda e um ícone de barra de busca à direita.

Fonte de criação própria

Figura 4.15 - Mensagem de erro, caso o CPF seja inválido



A imagem mostra uma interface de usuário com fundo escuro. No topo, há um ícone de menu hambúrguer à esquerda e uma mensagem de erro em vermelho: "O CPF é Inválido.". Abaixo, o título "Cadastrar um novo cartão de crédito" é exibido. O formulário contém campos para: "CPF do titular" (com o valor 34455488421), "número do cartão" (7755899300), "Código de segurança" (211), "Data de Validade" (20/23) e "Nome no cartão" (Gabriel L Bertazzi). Na base do formulário, há dois botões azuis: "Voltar" e "Cadastrar".

Fonte de criação própria

- Registro de endereços: Nesta parte do programa, são adquiridos os dados de endereço onde está localizado as garagens, para que ela seja colocada no mapa do aplicativo, nesta tela do aplicativo, existe um mapa pequeno na parte inferior com uma barra de procura, neste local o usuário coloca o endereço dele para que o local seja mostrado no mapa, fazendo isso o próprio mapa guarda todos os dados relacionados ao endereço que são necessários para que ele possa ser guardado no banco de dados. As outras informações presentes nesta parte do aplicativo, são o número de vagas que a pessoa tem disponível, além de o nome da garagem, que é necessário para diferenciar 2 vagas diferente em um mesmo endereço. A figura 4.16 mostra como ficou a área de registro de endereço.

Figura 4.16 - Tela de registro de endereço

Fonte de criação própria

- Data: Por fim a última parte do programa, trata-se de uma tela parecida com as telas anteriores, no qual existem algumas caixas de texto para que o usuário preencha com o dia, o mês, o ano, e a hora que a pessoa quer reservar uma vaga para ela. Apesar de parecer uma parte igual as outras, e bastante simples do programa, esta parte do aplicativo tem um controle de data bastante complexa, para que o usuário não selecione uma data inválida, ou uma data que já passou, dentro deste controle são levados em considerações datas possíveis, sendo que são olhados meses com 30 ou 31 ou 28 dias, além de olhar se o ano é bissexto para que a pessoa consiga selecionar a data no dia 29 de fevereiro nos anos bissextos, além de a pessoa não conseguir selecionar nem uma data que já passou nem um horário que já passou; este controle acabou ficando bastante grande na parte de código, o que pode ser visto no código do programa em anexo. A figura 4.17 mostra a tela da parte Data, e a figura 4.18 mostra a mensagem de erro caso o usuário escolha uma data inválida.

Figura 4.17 - Tela aonde o usuário vai selecionar a data da reserva

Escolha a data

dia

mes

ano

hora

Voltar Selecionar

Fonte de criação própria.

Figura 4.18 - Mensagem de erro caso a pessoa selecione uma data inválida.

Escolha a data

A data selecionada é inválida

dia

mes

ano

hora

Voltar Selecionar

Fonte de criação própria

4.2.Implementações Futuras.

4.2.1. Como encontrar vagas em locais públicos

Para encontrar vagas em locais públicos será utilizado a técnica de reconhecimento de imagem, em tempo real. Esta técnica se baseia na criação de um algoritmo que receberá as imagens de uma câmera de segurança de rua, e analisará esta imagem para encontrar locais próximos a calçada, nos quais seja possível estacionar um carro, além disso o programa será capaz de diferenciar vagas grandes de 2 vagas de carro, logo ele irá analisar o tamanho do espaço disponível separando as vagas possíveis em 3 tipos, pequenas, médias e grandes. Para que isto seja possível, será utilizado de um algoritmo de Deep Learning, que será treinado para reconhecer as imagens disponibilizadas para ele através de câmeras, e identificar os locais e vagas em cada rua.

Como o reconhecimento das vagas analisaram várias coisas ao mesmo tempo, como a ausência de veículos, o local (as vagas sempre devem estar próximas as calçadas), o tamanho disponível e a legalidade do local onde será estacionado o veículo, então serão necessário bem mais do que apenas um treino para o algoritmo, como também serão necessários provavelmente mais de um algoritmo. Por conta disso terão de ser criados várias bases de dados para a criação dos códigos, além de um servidor bastante potente, para que estes algoritmos de análise rodem recebendo várias imagens ao mesmo tempo e criando os resultados, tudo isso feito na nuvem, e o aplicativo só receberá os resultados advindos deste programa, e mostrará de forma gráfica no mapa.

Uma coisa importante de ser citada nesta parte do texto é que o servidor necessário para receber as imagens das câmeras e analisar essas imagens é um servidor que tem que ser muito forte, e a internet que será utilizada para que as imagens sejam enviadas para o servidor tem que ser uma internet também bastante boa, para que as imagens sejam enviadas com uma velocidade decente e o APP consiga informar bem se existem ou não vagas nos locais procurados.

4.2.2. Problemas que devem ser levados em consideração para a criação do algoritmo em Deep Learning.

Nesta parte do tópico serão levantados problemas que devem ser resolvidos, levados em consideração na criação do aplicativo, alguns deles já foram citados no tópico anterior, como por exemplo o tamanho da vaga. As figuras 4.19 e 4.20 mostram respectivamente, uma vaga com bastante espaço, e um espaço entre 2 carros que não cabe nenhum carro.

Figura 4.19 - Figura que mostra bastante espaço livre no qual claramente cabe um carro ou dois carros



Fonte: de criação própria.

Figura 4.20 - Espaço livre entre 2 carros, entretanto que não cabe um carro entre eles.



Fonte de criação própria.

É necessário que o software consiga reconhecer que na figura 2, não existe nenhuma vaga, pois, caso ele marque no aplicativo que exista uma vaga, ele vai enganar o seu usuário, o que irá causar uma decepção no mesmo, e a possível perda de um cliente.

Além dos espaços vazios perto das calçadas, o programa também tem que entender que alguns lugares não são permitidos estacionar, portanto ele não pode registrar estes locais como possíveis vagas no seu software. Um destes locais, são a frente de garagens, ou de locais com a guia rebaixada, como mostrado na figura 4.21.

Figura 4.21 - Local vago a frente de uma garagem, que não pode ser representado como vaga pelo aplicativo.



Fonte de criação própria.

O algoritmo também deve conseguir identificar também placas de trânsito relacionadas a estacionamento (figura 4.22) para saber se o local da imagem que está sendo passada para ele é permitido o estacionamento e/ou a parada. A figura 4.23 mostra um caso de um carro recebendo uma multa por estacionar em um local não permitido.

Figura 4.22 - Placas de lei de trânsito na ordem: Permitido estacionar e parar, proibido estacionar, proibido estacionar e parar



Fonte adaptado de <<https://aquitemplacas.com.br/produtosDetalhes.php?p=de-estacionamento>>

Figura 4.23 - Figura de um carro sendo advertido por desobedecer a placa de trânsito



Fonte adaptado de <<http://www.maceio.al.gov.br/2016/11/saiba-como-obter-desconto-no-pagamento-de-multas/>>

É importante tomar estes cuidados, pois caso o aplicativo informe que existe uma vaga em um local no qual não é permitido estacionar, isso faz com que o cliente se dirija ao local em vão, logo isto demonstra uma falha, pois o aplicativo que deveria ajudar o usuário a encontrar uma vaga de forma mais rápida, está na verdade fazendo ele ir a lugares não encontrar nenhuma local para estacionar e se frustrar.

Uma outra coisa importante de se pensar, são as faixas de estacionamento, no caso a faixa amarela ou branca contínua perto da calçada que significa que não pode estacionar naquele local e a faixa branca tracejada que significa que se pode estacionar naquele local, entretanto existem faixas brancas em alguns lugares que são colocadas na transversal para delimitar o tamanho de cada vaga próximo a calçada e o aplicativo além de saber das diferenças das linhas próximas a calçada deve entender que as linhas na transversal não influenciam na permissão de estacionar. Um problema além destes é que nem todos os lugares tem estas faixas próximas a guia, entretanto isto acontece por uma falha, ou

descuido do governo, pois nas leis de sinalização de trânsito elas são citadas, e devem ser obedecidas, logo o aplicativo deve levar isso em consideração, e caso não haja as faixas deve-se observar outras possíveis sinalizações de trânsito, como placas ou garagens, e caso não haja nenhuma, o aplicativo pode dizer que pode-se estacionar naquele local, desde que o local seja um bairro residencial ou comercial, e não se encontre em uma grande avenida. As figuras 4.24 e 4.25 mostram as faixas amarela contínua e branca tracejada.

Figura 4.24 - Faixa amarela que diz que é proibido estacionar.



Fonte adaptado de <<https://infonet.com.br/noticias/cidade/proibicao-de-estacionamento-divide-opinioes-de-comerciantes/>>

Figura 4.25- imagem que mostra linhas tracejadas brancas que permitem que o carro estacione, além de mostrar uma delimitação feita com linhas para cada vaga.



Fonte adaptado de <<https://portallitoralsul.com.br/demutran-orienta-os-motoristas-respeitarem-os-limites-das-vagas-na-avenida-25-de-julho/estacionamento-demutran-3/>>

Por fim a última regra que falta que deve ser implementada no aplicativo para que não gere problemas para o usuário é a lei de não estacionar próximo ao canteiro central. Muitas avenidas, tem um canteiro que as divide em lado que vai e lado que volta, este canteiro existe para aumentar a segurança das pessoas que estão dirigindo, visto que com ele ali, nem uma pessoa fará uma ultrapassagem pela contramão. Como ele existe por este motivo, além de ser esteticamente bonito também, não existe um espaço próximo ao canteiro central para que pessoas estacionem ali, logo é proibido por lei estacionar próximo aos canteiros centrais, então o aplicativo que encontrará vagas em locais público deve conseguir identificar um canteiro central e saber que naquele local é proibido estacionar. A figura 4.26 mostra a foto de um canteiro central que deve ser reconhecido pelo aplicativo.

Figura 4.26 - Foto de um canteiro central que o aplicativo deve reconhecer.



Fonte adaptado de <<http://semmas.manaus.am.gov.br/canteiro-central-da-torquato-tapajos-recebe-mudas-da-prefeitura-de-manaus/>>

Estas leis de trânsito devem ser lembradas para que o restante do aplicativo seja executado de forma adequada sem causar problemas aos usuários. O que ficou claro nesta parte do trabalho é que o algoritmo de deep learning que deve ser gerado para este aplicativo será bastante complexo, visto que ele deverá passar por vários treinos aprendendo cada uma das regras de trânsito citadas neste tópico, o que implica em vários bancos de dados diferentes (1 para cada regra) com as imagens de treino que serão utilizadas pelo aplicativo. O que deixa claro que o aplicativo final é bastante complexo de ser executado, mas também deixa claro que ele é possível de ser executado, visto que nenhuma das regras é impossível de ser apreendida.

5. Conclusão.

A primeira ferramenta da administração citada, o modelo canvas, foi estudada e aplicada de forma a criar um modelo de negócio bastante interessante, e coerente, que tem alta probabilidade de aplicação no mercado atual, visto que, a pesquisa de mercado

não mostrou resultados bastante favoráveis ao modelo, e será comentada no próximo parágrafo.

Já sobre a pesquisa de mercado foi visto como deve ser a comunicação da empresa com os seus clientes, observando as respostas de perfil feitas na primeira seção do formulário desenvolvido. Além disso a terceira seção dá uma segurança que o aplicativo seria aceito pelo mercado, logo ele legitima o modelo de negócios, ou seja, faz com que este tenha um maior valor e mostra segurança para que o projeto seja desenvolvido. E por fim a quinta seção fez com que uma das estruturas de receita do modelo de negócios tivesse de ser alterada, fazendo com que o modelo ficasse mais fiel a realidade dos clientes e deixando Canvas ainda mais robusto, logo, a pesquisa de mercado foi de suma importância para a validação do projeto.

Após a legitimação do modelo de negócios, foi desenvolvido o protótipo do aplicativo, como proposto nos objetivos. O que ficou claro é que este modelo criado do produto precisa de um polimento maior antes de ser entregue ao mercado, mas ele também mostra que o aplicativo é possível de ser criado, e que ele funcionou muito bem, apesar de aparecerem alguns erros com caracteres especiais. Entendendo estes problemas deve-se trabalhar para que eles sejam resolvidos antes do produto entrar no mercado de trabalho, dando foco principal na melhoria da experiência do usuário.

Sobre a continuação do aplicativo e as implementações de como deve ser criado o algoritmo para verificação em vias públicas foram mostradas no texto no tópico 4.2. A partir das informações deste tópico ficou claro as dificuldades desta aplicação, entretanto também ficou claro que ela é possível, o que é recomendado, é que seja feito um estudo mais aprofundado na área de Deep Learning e verificar a viabilidade de preço de ter um servidor que receba as imagens das câmeras para o processamento de dados dela e ver se esse valor é possível de ser pago pela utilização do APP pelos usuários. Em um primeiro momento é recomendado que o aplicativo fique somente com a parte de verificação em lugares privados.

Em resumo, o projeto tem uma boa aderência de mercado, pois ele trabalha em cima de um problema que grande parte da população sofre que é a falta de vagas e a pesquisa de mercado provou isso. Portanto o aplicativo é um produto bastante viável e que teriam custos pequenos no início se o plano de negócio estipulado começasse pequeno e com poucas pessoas trabalhando. Para que o produto tenha uma boa aderência no mercado é necessário que seja trabalhado a estética e a experiência do usuário, e então com estes pequenos ajustes o projeto poderá ser lançado no mercado e gradualmente irá

evoluir até que no futuro talvez seja possível que ele encontre vagas em lugares públicos e privados.

6. Bibliografia

[1] Lecum, Yann; Bengio, Yoshua; Hinton, Geoffrey; “Deep Learning”, Macmillian Publisher Limited, 2015. Visualizado dia 10 de novembro de 2018

[2] Canal 3blue1brown, ‘Mas o que *é* uma rede neural? “Deep Learning” cap. 1’: <<https://www.youtube.com/watch?v=aircAruvnKk&t=577s>> Visualizado dia 15 de novembro de 2018

[3] Canal 3blue1brown, descida de gradiente, como as redes neurais aprendem | capítulo 2 “Deep Learning”: <<https://www.youtube.com/watch?v=IHZwWFHwa-w&t=689s>> visualizado dia 15 de novembro de 2018

[4] Wikipédia, Paradigma de programação: <https://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o> Visualizado dia 18 de março de 2019

[5] Oficina da Net, “Qual a diferença entre programação estruturada e programação orientada a objetos?”: <<https://www.oficinadanet.com.br/post/14463-qual-a-diferenca-entre-programacao-estruturada-e-programacao-orientada-a-objetos>> Visualizado dia 18 de março de 2019

[6] Wikipédia, Paradigma programação imperativa: <https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_imperativa> Visualizado dia 18 de março de 2019

[7] Webholic, Google app inventor: A programação visual para aplicativos móveis ganh a um reforço de peso: <<http://webholic.com.br/google-app-inventor-a-programacao-visual-para-equipamentos-moveis-ganha-um-reforco-de-peso/>> Visualizado dia 18 de março de 2019

[8] Wikipédia, Paradigma programação genérica: <https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_gen%C3%A9rica> Visualizado dia 18 de março de 2019

[9] Wikipédia, C++: <<https://pt.wikipedia.org/wiki/C%2B%2B>> Visualizado dia 18 de março de 2019

[10] Wikipédia, java linguagem de programação: <[http://pt.wikipedia.org/Java_\(linguagem_de_programa%C3%A7%C3%A3o\)](http://pt.wikipedia.org/Java_(linguagem_de_programa%C3%A7%C3%A3o))> Visualizado dia 18 de março de 2019

[11] Wikipédia, lua linguagem de programação: <[https://pt.wikipedia.org/wiki/Lua_\(linguagem_de_programa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/Lua_(linguagem_de_programa%C3%A7%C3%A3o))> Visualizado dia 18 de março de 2019

[12] Wikipédia, Corona Software: <[https://en.wikipedia.org/wiki/Corona_\(software\)](https://en.wikipedia.org/wiki/Corona_(software))> Visualizado dia 18 de março de 2019

[13] Wikipédia, App Inventor: <https://pt.wikipedia.org/wiki/App_Inventor> Visualizado dia 18 de março de 2019

[14] Appinventor, About us: <<http://appinventor.mit.edu/explore/about-us.html>> Visualizado dia 18 de março de 2019

[15] SEBRAE, Canvas: como estruturar o seu negócio de empresa: <<https://www.sebraepr.com.br/como-estruturar-seu-modelo-de-negocio/>> Visualizado dia 10 de abril de 2019

[16] Ministério da Infraestrutura, como funciona a matriz GUT: <<http://portaldaestrategia.infraestrutura.gov.br/ultimas-noticias/282-como-funciona-a-matriz-gut.html>> Visualizado dia 10 de abril de 2019

[17] Grando. Nei, Explicando o modelo de negócio (Canvas): < <https://pt.slideshare.net/neigrando/explicando-o-modelo-de-negcios-viradaempreendedora>> Visualizado dia 10 de abriu de 2019

[18] Moraes. Daniel, Pesquisa de mercado, o que é como fazer os principais tipos: <<https://rockcontent.com/blog/pesquisa-de-mercado/>> Visualizado dia 15 de outubro de 2018.

7. Anexos.

7.1. Programas feitos no Corona (linguagem lua)

Main.lua:

```

1  local composer = require("composer")
2  local scene = composer.newScene()
3  local localGroup = display.newGroup()
4  local widget = require("widget")
5  local json = require("json")
6  local sqlite3 = require("sqlite3")
7
8
9  W = display.contentWidth
10 H = display.contentHeight
11 LoginNumber = 1
12 CPF = 1
13
14
15 dateh = tonumber(os.date( "%H" ))
16 dated = tonumber(os.date( "%d" ))
17 datem = tonumber(os.date( "%m" ))
18 datey = tonumber(os.date( "%Y" ))
19 datadia = tonumber(os.date( "%d" ))
20 datames = tonumber(os.date( "%m" ))
21 dataano = tonumber(os.date( "%Y" ))
22
23 datahora = tonumber(dateh)+1
24
25 longitudedagaragem = ""
26 latitudedagaragem = ""
27 iddagaragem = 1
28
29
30 Interface = display.newRect(0,0,0,0)
31 Interface.width = W;Interface.height = 200
32 Interface.anchorX = 0; Interface.anchorY = 0
33
34 local numeros = {}
35 for i=1,99 do
36     table.insert(numeros,i)
37 end
38
39 -----Botão de procurar data-----
40 botaoData = widget.newButton{
41     width = 163,
42     height = 54,
43     defaultFile = "Button.png",
44     overFile = "Button.png",
45     label = datahora..":00".."\n"..datadia.."/"..datames.."/"..dataano,    --datadia.."/"..datames.."/"..dataano
46     fontSize = 22,
47     labelColor = {default = {1,1,1}},

```

```

48     emboss=false,
49     onRelease = Data_function,
50     x = 82,
51     y = 170,
52 }
53 botaoData.isVisible = true
54
55
56
57 local function atualizarData()
58     botaoData:setLabel(datahora..":00".."\n"..datadia.."/"..datames.."/"..dataano)
59 end
60 Runtime:addEventListener("enterFrame",atualizarData)
61 -----
62 -- PICKER WHEEL -----
63 columnData = {
64     {
65         width = 150,
66         startIndex = 2,
67         labelPadding = 10,
68         labels = {"pequeno", "medio", "grande", "moto"},
69     },
70     {
71         width = 100,
72         startIndex = 2,
73         labelPadding = 0,
74         labels = numeros,
75     }
76 }
77 -- Create a new Picker Wheel
78 pickerWheel = widget.newPickerWheel {
79     top = 50,
80     left = 200,
81     width = 250,
82     columns = columnData,
83     rowHeight = 40,
84     style = "resizable",
85     fontSize = 35
86 }
87
88 -----
89 -- botao de reservar -----
90 local function ReservarVaga( event )
91
92     if ( "ended" == event.phase ) then
93         if iddagaragem ~= 0 then

```

```

94 local texto = ""
95 local valores = pickerWheel:getValues()
96 local tipo = valores[1].value
97 local numero = valores[2].value
98 print(tipo,numero)
99 for row in db:nrows("SELECT * FROM enderecos WHERE id=" .. idgaragem) do
100     texto = "Garagem: "..row.nome.."\\n"
101     texto = texto.."Endereço:"..row.endereco.."\\n"
102     texto = texto.."VAGAS\\nTipo: "..tipo.."\\nnumero "..numero.."\\n"
103     texto = texto.."Data: "..datadia.."/"..datames.."/"..dataano
104     texto = texto.."horário: "..datahora
105 end
106 local function cadastrarNaDB()
107     local function cadastroReserva( event )
108         if ( event.isError ) then
109             local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )
110             print(event.response)
111         else
112             if event.response == "Registrado com sucesso ." then
113                 local alert = native.showAlert( "Cadastrando no banco de dados", "Resultado: "..event.response, { "Ok" } )
114             else
115                 local alert = native.showAlert( "Erro ao Cadastrar", event.response, { "Try again" } )
116             end
117         end
118     end
119     parameters = {}
120     parameters.body = "Register=1&dia=" .. datadia .. "&mes=" .. datames .. "&ano=" .. dataano .. "&hora=".. datahora..
121     "&tipo="..tipo.."&numero=".. numero .. "&idGaragem=".. idgaragem .. "&CPF=" .. CPF
122
123     local URL2 = "http://www.coronasdkgames.com/Gabriel/cadastroReserva.php"
124     network.request(URL2, "POST", cadastroReserva,parameters) -- baixando base de dados completa
125 end
126 native.showAlert("Confirme Sua Reserva!",texto,{"cancelar","Confirmar"}, cadastrarNaDB)
127 else
128     native.showAlert("Atenção!", "Selecione uma garagem para fazer a reserva!",{"OK"})
129 end
130 end
131 end
132
133 botaoReserva = widget.newButton(
134 {
135     width = 193,
136     height = 64,
137     defaultFile = "reservarUp.png",
138     overFile = "reservarDown.png",
139     onEvent = ReservarVaga
140 }
141 )
142 botaoReserva.anchorX = 1
143 botaoReserva.anchorY = 0
144 botaoReserva.x = W - 10
145 botaoReserva.y = 110

```

```

148 button = display.newImage("Menu.png",0,0)
149 button.anchorX=0
150 button.anchorY=0
151 button.x = 0
152 button.y = 0
153
154 -- LIMPAP DOCUMENTS --
155 local lfs = require "lfs";
156
157 local doc_dir = system.DocumentsDirectory;
158 local doc_path = system.pathForFile("", doc_dir);
159 local resultOK, errorMsg;
160
161 for file in lfs.dir(doc_path) do
162     local theFile = system.pathForFile(file, doc_dir);
163
164     if (lfs.attributes(theFile, "mode") ~= "directory") then
165         resultOK, errorMsg = os.remove(theFile);
166
167         if (resultOK) then
168             print(file.." removed");
169         else
170             print("Error removing file: "..file.." "..errorMsg);
171         end
172     end
173 end
174
175 -----
176
177 myMap = native.newMapView(0, 0, display.contentWidth, display.contentHeight-100)
178 myMap.mapType = "hybrid"
179 local attempts = 0
180 myMap.anchorX,myMap.anchorY=0,0
181 myMap.x = 0
182 myMap.y = 200
183 myMap.isVisible = true
184 local function locationHandler( event )
185
186     local currentLocation = myMap.getUserLocation()
187
188     if ( currentLocation.errorCode or ( currentLocation.latitude == 0 and currentLocation.longitude == 0 ) ) then
189
190         attempts = attempts + 1
191
192         if ( attempts > 10 ) then
193             native.showAlert( "No GPS Signal", "Can't sync with GPS.", { "Okay" } )
194         else
195             timer.performWithDelay( 1000, locationHandler )
196         end
197     else
198         local function markerListener(event)
199             print( "type: ", event.type ) -- event type
200             print( "lat: ", event.latitude )
201             print( "lon: ", event.longitude )
202             print( "alt: ", event.altitude )
203             print( "acc: ", event.accuracy )
204             print( "speed: ", event.speed )
205             print( "heading: ", event.heading )
206             print( "magneticHeading: ", event.magneticHeading )
207             print( "trueHeading: ", event.trueHeading )
208             print( "pitch: ", event.pitch )
209             print( "roll: ", event.roll )
210             print( "yaw: ", event.yaw )
211             print( "x: ", event.x )
212             print( "y: ", event.y )
213             print( "z: ", event.z )
214             print( "x2: ", event.x2 )
215             print( "y2: ", event.y2 )
216             print( "z2: ", event.z2 )
217             print( "x3: ", event.x3 )
218             print( "y3: ", event.y3 )
219             print( "z3: ", event.z3 )
220             print( "x4: ", event.x4 )
221             print( "y4: ", event.y4 )
222             print( "z4: ", event.z4 )
223             print( "x5: ", event.x5 )
224             print( "y5: ", event.y5 )
225             print( "z5: ", event.z5 )
226             print( "x6: ", event.x6 )
227             print( "y6: ", event.y6 )
228             print( "z6: ", event.z6 )
229             print( "x7: ", event.x7 )
230             print( "y7: ", event.y7 )
231             print( "z7: ", event.z7 )
232             print( "x8: ", event.x8 )
233             print( "y8: ", event.y8 )
234             print( "z8: ", event.z8 )
235             print( "x9: ", event.x9 )
236             print( "y9: ", event.y9 )
237             print( "z9: ", event.z9 )
238             print( "x10: ", event.x10 )
239             print( "y10: ", event.y10 )
240             print( "z10: ", event.z10 )
241             print( "x11: ", event.x11 )
242             print( "y11: ", event.y11 )
243             print( "z11: ", event.z11 )
244             print( "x12: ", event.x12 )
245             print( "y12: ", event.y12 )
246             print( "z12: ", event.z12 )
247             print( "x13: ", event.x13 )
248             print( "y13: ", event.y13 )
249             print( "z13: ", event.z13 )
250             print( "x14: ", event.x14 )
251             print( "y14: ", event.y14 )
252             print( "z14: ", event.z14 )
253             print( "x15: ", event.x15 )
254             print( "y15: ", event.y15 )
255             print( "z15: ", event.z15 )
256             print( "x16: ", event.x16 )
257             print( "y16: ", event.y16 )
258             print( "z16: ", event.z16 )
259             print( "x17: ", event.x17 )
260             print( "y17: ", event.y17 )
261             print( "z17: ", event.z17 )
262             print( "x18: ", event.x18 )
263             print( "y18: ", event.y18 )
264             print( "z18: ", event.z18 )
265             print( "x19: ", event.x19 )
266             print( "y19: ", event.y19 )
267             print( "z19: ", event.z19 )
268             print( "x20: ", event.x20 )
269             print( "y20: ", event.y20 )
270             print( "z20: ", event.z20 )
271             print( "x21: ", event.x21 )
272             print( "y21: ", event.y21 )
273             print( "z21: ", event.z21 )
274             print( "x22: ", event.x22 )
275             print( "y22: ", event.y22 )
276             print( "z22: ", event.z22 )
277             print( "x23: ", event.x23 )
278             print( "y23: ", event.y23 )
279             print( "z23: ", event.z23 )
280             print( "x24: ", event.x24 )
281             print( "y24: ", event.y24 )
282             print( "z24: ", event.z24 )
283             print( "x25: ", event.x25 )
284             print( "y25: ", event.y25 )
285             print( "z25: ", event.z25 )
286             print( "x26: ", event.x26 )
287             print( "y26: ", event.y26 )
288             print( "z26: ", event.z26 )
289             print( "x27: ", event.x27 )
290             print( "y27: ", event.y27 )
291             print( "z27: ", event.z27 )
292             print( "x28: ", event.x28 )
293             print( "y28: ", event.y28 )
294             print( "z28: ", event.z28 )
295             print( "x29: ", event.x29 )
296             print( "y29: ", event.y29 )
297             print( "z29: ", event.z29 )
298             print( "x30: ", event.x30 )
299             print( "y30: ", event.y30 )
300             print( "z30: ", event.z30 )
301             print( "x31: ", event.x31 )
302             print( "y31: ", event.y31 )
303             print( "z31: ", event.z31 )
304             print( "x32: ", event.x32 )
305             print( "y32: ", event.y32 )
306             print( "z32: ", event.z32 )
307             print( "x33: ", event.x33 )
308             print( "y33: ", event.y33 )
309             print( "z33: ", event.z33 )
310             print( "x34: ", event.x34 )
311             print( "y34: ", event.y34 )
312             print( "z34: ", event.z34 )
313             print( "x35: ", event.x35 )
314             print( "y35: ", event.y35 )
315             print( "z35: ", event.z35 )
316             print( "x36: ", event.x36 )
317             print( "y36: ", event.y36 )
318             print( "z36: ", event.z36 )
319             print( "x37: ", event.x37 )
320             print( "y37: ", event.y37 )
321             print( "z37: ", event.z37 )
322             print( "x38: ", event.x38 )
323             print( "y38: ", event.y38 )
324             print( "z38: ", event.z38 )
325             print( "x39: ", event.x39 )
326             print( "y39: ", event.y39 )
327             print( "z39: ", event.z39 )
328             print( "x40: ", event.x40 )
329             print( "y40: ", event.y40 )
330             print( "z40: ", event.z40 )
331             print( "x41: ", event.x41 )
332             print( "y41: ", event.y41 )
333             print( "z41: ", event.z41 )
334             print( "x42: ", event.x42 )
335             print( "y42: ", event.y42 )
336             print( "z42: ", event.z42 )
337             print( "x43: ", event.x43 )
338             print( "y43: ", event.y43 )
339             print( "z43: ", event.z43 )
340             print( "x44: ", event.x44 )
341             print( "y44: ", event.y44 )
342             print( "z44: ", event.z44 )
343             print( "x45: ", event.x45 )
344             print( "y45: ", event.y45 )
345             print( "z45: ", event.z45 )
346             print( "x46: ", event.x46 )
347             print( "y46: ", event.y46 )
348             print( "z46: ", event.z46 )
349             print( "x47: ", event.x47 )
350             print( "y47: ", event.y47 )
351             print( "z47: ", event.z47 )
352             print( "x48: ", event.x48 )
353             print( "y48: ", event.y48 )
354             print( "z48: ", event.z48 )
355             print( "x49: ", event.x49 )
356             print( "y49: ", event.y49 )
357             print( "z49: ", event.z49 )
358             print( "x50: ", event.x50 )
359             print( "y50: ", event.y50 )
360             print( "z50: ", event.z50 )
361             print( "x51: ", event.x51 )
362             print( "y51: ", event.y51 )
363             print( "z51: ", event.z51 )
364             print( "x52: ", event.x52 )
365             print( "y52: ", event.y52 )
366             print( "z52: ", event.z52 )
367             print( "x53: ", event.x53 )
368             print( "y53: ", event.y53 )
369             print( "z53: ", event.z53 )
370             print( "x54: ", event.x54 )
371             print( "y54: ", event.y54 )
372             print( "z54: ", event.z54 )
373             print( "x55: ", event.x55 )
374             print( "y55: ", event.y55 )
375             print( "z55: ", event.z55 )
376             print( "x56: ", event.x56 )
377             print( "y56: ", event.y56 )
378             print( "z56: ", event.z56 )
379             print( "x57: ", event.x57 )
380             print( "y57: ", event.y57 )
381             print( "z57: ", event.z57 )
382             print( "x58: ", event.x58 )
383             print( "y58: ", event.y58 )
384             print( "z58: ", event.z58 )
385             print( "x59: ", event.x59 )
386             print( "y59: ", event.y59 )
387             print( "z59: ", event.z59 )
388             print( "x60: ", event.x60 )
389             print( "y60: ", event.y60 )
390             print( "z60: ", event.z60 )
391             print( "x61: ", event.x61 )
392             print( "y61: ", event.y61 )
393             print( "z61: ", event.z61 )
394             print( "x62: ", event.x62 )
395             print( "y62: ", event.y62 )
396             print( "z62: ", event.z62 )
397             print( "x63: ", event.x63 )
398             print( "y63: ", event.y63 )
399             print( "z63: ", event.z63 )
400             print( "x64: ", event.x64 )
401             print( "y64: ", event.y64 )
402             print( "z64: ", event.z64 )
403             print( "x65: ", event.x65 )
404             print( "y65: ", event.y65 )
405             print( "z65: ", event.z65 )
406             print( "x66: ", event.x66 )
407             print( "y66: ", event.y66 )
408             print( "z66: ", event.z66 )
409             print( "x67: ", event.x67 )
410             print( "y67: ", event.y67 )
411             print( "z67: ", event.z67 )
412             print( "x68: ", event.x68 )
413             print( "y68: ", event.y68 )
414             print( "z68: ", event.z68 )
415             print( "x69: ", event.x69 )
416             print( "y69: ", event.y69 )
417             print( "z69: ", event.z69 )
418             print( "x70: ", event.x70 )
419             print( "y70: ", event.y70 )
420             print( "z70: ", event.z70 )
421             print( "x71: ", event.x71 )
422             print( "y71: ", event.y71 )
423             print( "z71: ", event.z71 )
424             print( "x72: ", event.x72 )
425             print( "y72: ", event.y72 )
426             print( "z72: ", event.z72 )
427             print( "x73: ", event.x73 )
428             print( "y73: ", event.y73 )
429             print( "z73: ", event.z73 )
430             print( "x74: ", event.x74 )
431             print( "y74: ", event.y74 )
432             print( "z74: ", event.z74 )
433             print( "x75: ", event.x75 )
434             print( "y75: ", event.y75 )
435             print( "z75: ", event.z75 )
436             print( "x76: ", event.x76 )
437             print( "y76: ", event.y76 )
438             print( "z76: ", event.z76 )
439             print( "x77: ", event.x77 )
440             print( "y77: ", event.y77 )
441             print( "z77: ", event.z77 )
442             print( "x78: ", event.x78 )
443             print( "y78: ", event.y78 )
444             print( "z78: ", event.z78 )
445             print( "x79: ", event.x79 )
446             print( "y79: ", event.y79 )
447             print( "z79: ", event.z79 )
448             print( "x80: ", event.x80 )
449             print( "y80: ", event.y80 )
450             print( "z80: ", event.z80 )
451             print( "x81: ", event.x81 )
452             print( "y81: ", event.y81 )
453             print( "z81: ", event.z81 )
454             print( "x82: ", event.x82 )
455             print( "y82: ", event.y82 )
456             print( "z82: ", event.z82 )
457             print( "x83: ", event.x83 )
458             print( "y83: ", event.y83 )
459             print( "z83: ", event.z83 )
460             print( "x84: ", event.x84 )
461             print( "y84: ", event.y84 )
462             print( "z84: ", event.z84 )
463             print( "x85: ", event.x85 )
464             print( "y85: ", event.y85 )
465             print( "z85: ", event.z85 )
466             print( "x86: ", event.x86 )
467             print( "y86: ", event.y86 )
468             print( "z86: ", event.z86 )
469             print( "x87: ", event.x87 )
470             print( "y87: ", event.y87 )
471             print( "z87: ", event.z87 )
472             print( "x88: ", event.x88 )
473             print( "y88: ", event.y88 )
474             print( "z88: ", event.z88 )
475             print( "x89: ", event.x89 )
476             print( "y89: ", event.y89 )
477             print( "z89: ", event.z89 )
478             print( "x90: ", event.x90 )
479             print( "y90: ", event.y90 )
480             print( "z90: ", event.z90 )
481             print( "x91: ", event.x91 )
482             print( "y91: ", event.y91 )
483             print( "z91: ", event.z91 )
484             print( "x92: ", event.x92 )
485             print( "y92: ", event.y92 )
486             print( "z92: ", event.z92 )
487             print( "x93: ", event.x93 )
488             print( "y93: ", event.y93 )
489             print( "z93: ", event.z93 )
490             print( "x94: ", event.x94 )
491             print( "y94: ", event.y94 )
492             print( "z94: ", event.z94 )
493             print( "x95: ", event.x95 )
494             print( "y95: ", event.y95 )
495             print( "z95: ", event.z95 )
496             print( "x96: ", event.x96 )
497             print( "y96: ", event.y96 )
498             print( "z96: ", event.z96 )
499             print( "x97: ", event.x97 )
500             print( "y97: ", event.y97 )
501             print( "z97: ", event.z97 )
502             print( "x98: ", event.x98 )
503             print( "y98: ", event.y98 )
504             print( "z98: ", event.z98 )
505             print( "x99: ", event.x99 )
506             print( "y99: ", event.y99 )
507             print( "z99: ", event.z99 )
508             print( "x100: ", event.x100 )
509             print( "y100: ", event.y100 )
510             print( "z100: ", event.z100 )
511             print( "x101: ", event.x101 )
512             print( "y101: ", event.y101 )
513             print( "z101: ", event.z101 )
514             print( "x102: ", event.x102 )
515             print( "y102: ", event.y102 )
516             print( "z102: ", event.z102 )
517             print( "x103: ", event.x103 )
518             print( "y103: ", event.y103 )
519             print( "z103: ", event.z103 )
520             print( "x104: ", event.x104 )
521             print( "y104: ", event.y104 )
522             print( "z104: ", event.z104 )
523             print( "x105: ", event.x105 )
524             print( "y105: ", event.y105 )
525             print( "z105: ", event.z105 )
526             print( "x106: ", event.x106 )
527             print( "y106: ", event.y106 )
528             print( "z106: ", event.z106 )
529             print( "x107: ", event.x107 )
530             print( "y107: ", event.y107 )
531             print( "z107: ", event.z107 )
532             print( "x108: ", event.x108 )
533             print( "y108: ", event.y108 )
534             print( "z108: ", event.z108 )
535             print( "x109: ", event.x109 )
536             print( "y109: ", event.y109 )
537             print( "z109: ", event.z109 )
538             print( "x110: ", event.x110 )
539             print( "y110: ", event.y110 )
540             print( "z110: ", event.z110 )
541             print( "x111: ", event.x111 )
542             print( "y111: ", event.y111 )
543             print( "z111: ", event.z111 )
544             print( "x112: ", event.x112 )
545             print( "y112: ", event.y112 )
546             print( "z112: ", event.z112 )
547             print( "x113: ", event.x113 )
548             print( "y113: ", event.y113 )
549             print( "z113: ", event.z113 )
550             print( "x114: ", event.x114 )
551             print( "y114: ", event.y114 )
552             print( "z114: ", event.z114 )
553             print( "x115: ", event.x115 )
554             print( "y115: ", event.y115 )
555             print( "z115: ", event.z115 )
556             print( "x116: ", event.x116 )
557             print( "y116: ", event.y116 )
558             print( "z116: ", event.z116 )
559             print( "x117: ", event.x117 )
560             print( "y117: ", event.y117 )
561             print( "z117: ", event.z117 )
562             print( "x118: ", event.x118 )
563             print( "y118: ", event.y118 )
564             print( "z118: ", event.z118 )
565             print( "x119: ", event.x119 )
566             print( "y119: ", event.y119 )
567             print( "z119: ", event.z119 )
568             print( "x120: ", event.x120 )
569             print( "y120: ", event.y120 )
570             print( "z120: ", event.z120 )
571             print( "x121: ", event.x121 )
572             print( "y121: ", event.y121 )
573             print( "z121: ", event.z121 )
574             print( "x122: ", event.x122 )
575             print( "y122: ", event.y122 )
576             print( "z122: ", event.z122 )
577             print( "x123: ", event.x123 )
578             print( "y123: ", event.y123 )
579             print( "z123: ", event.z123 )
580             print( "x124: ", event.x124 )
581             print( "y124: ", event.y124 )
582             print( "z124: ", event.z124 )
583             print( "x125: ", event.x125 )
584             print( "y125: ", event.y125 )
585             print( "z125: ", event.z125 )
586             print( "x126: ", event.x126 )
587             print( "y126: ", event.y126 )
588             print( "z126: ", event.z126 )
589             print( "x127: ", event.x127 )
590             print( "y127: ", event.y127 )
591             print( "z127: ", event.z127 )
592             print( "x128: ", event.x128 )
593             print( "y128: ", event.y128 )
594             print( "z128: ", event.z128 )
595             print( "x129: ", event.x129 )
596             print( "y129: ", event.y129 )
597             print( "z129: ", event.z129 )
598             print( "x130: ", event.x130 )
599             print( "y130: ", event.y130 )
600             print( "z130: ", event.z130 )
601             print( "x131: ", event.x131 )
602             print( "y131: ", event.y131 )
603             print( "z131: ", event.z131 )
604             print( "x132: ", event.x132 )
605             print( "y132: ", event.y132 )
606             print( "z132: ", event.z132 )
607             print( "x133: ", event.x133 )
608             print( "y133: ", event.y133 )
609             print( "z133: ", event.z133 )
610             print( "x134: ", event.x134 )
611             print( "y134: ", event.y134 )
612             print( "z134: ", event.z134 )
613             print( "x135: ", event.x135 )
614             print( "y135: ", event.y135 )
615             print( "z135: ", event.z135 )
616             print( "x136: ", event.x136 )
617             print( "y136: ", event.y136 )
618             print( "z136: ", event.z136 )
619             print( "x137: ", event.x137 )
620             print( "y137: ", event.y137 )
621             print( "z137: ", event.z137 )
622             print( "x138: ", event.x138 )
623             print( "y138: ", event.y138 )
624             print( "z138: ", event.z138 )
625             print( "x139: ", event.x139 )
626             print( "y139: ", event.y139 )
627             print( "z139: ", event.z139 )
628             print( "x140: ", event.x140 )
629             print( "y140: ", event.y140 )
630             print( "z140: ", event.z140 )
631             print( "x141: ", event.x141 )
632             print( "y141: ", event.y141 )
633             print( "z141: ", event.z141 )
634             print( "x142: ", event.x142 )
635             print( "y142: ", event.y142 )
636             print( "z142: ", event.z142 )
637             print( "x143: ", event.x143 )
638             print( "y143: ", event.y143 )
639             print( "z143: ", event.z143 )
640             print( "x144: ", event.x144 )
641             print( "y144: ", event.y144 )
642             print( "z144: ", event.z144 )
643             print( "x145: ", event.x145 )
644             print( "y145: ", event.y145 )
645             print( "z145: ", event.z145 )
646             print( "x146: ", event.x146 )
647             print( "y146: ", event.y146 )
648             print( "z146: ", event.z146 )
649             print( "x147: ", event.x147 )
650             print( "y147: ", event.y147 )
651             print( "z147: ", event.z147 )
652             print( "x148: ", event.x148 )
653             print( "y148: ", event.y148 )
654             print( "z148: ", event.z148 )
655             print( "x149: ", event.x149 )
656             print( "y149: ", event.y149 )
657             print( "z149: ", event.z149 )
658             print( "x150: ", event.x150 )
659             print( "y150: ", event.y150 )
660             print( "z150: ", event.z150 )
661             print( "x151: ", event.x151 )
662             print( "y151: ", event.y151 )
663             print( "z151: ", event.z151 )
664             print( "x152: ", event.x152 )
665             print( "y152: ", event.y152 )
666             print( "z152: ", event.z152 )
667             print( "x153: ", event.x153 )
668             print( "y153: ", event.y153 )
669             print( "z153: ", event.z153 )
670             print( "x154: ", event.x154 )
671             print( "y154: ", event.y154 )
672             print( "z154: ", event.z154 )
673             print( "x155: ", event.x155 )
674             print( "y155: ", event.y155 )
675             print( "z155: ", event.z155 )
676             print( "x156: ", event.x156 )
677             print( "y156: ", event.y156 )
678             print( "z156: ", event.z156 )
679             print( "x157: ", event.x157 )
680             print( "y157: ", event.y157 )
681             print( "z157: ", event.z157 )
682             print( "x158: ", event.x158 )
683             print( "y158: ", event.y158 )
684             print( "z158: ", event.z158 )
685             print( "x159: ", event.x159 )
686             print( "y159: ", event.y159 )
687             print( "z159: ", event.z159 )
688             print( "x160: ", event.x160 )
689             print( "y160: ", event.y160 )
690             print( "z160: ", event.z160 )
691             print( "x161: ", event.x161 )
692             print( "y161: ", event.y161 )
693             print( "z161: ", event.z161 )
694             print( "x162: ", event.x162 )
695             print( "y162: ", event.y162 )
696             print( "z162: ", event.z162 )
697             print( "x163: ", event.x163 )
698             print( "y163: ", event.y163 )
699             print( "z163: ", event.z163 )
700             print( "x164: ", event.x164 )
701             print( "y164: ", event.y164 )
702             print( "z164: ", event.z164 )
703             print( "x165: ", event.x165 )
704             print( "y165: ", event.y165 )
705             print( "z165: ", event.z165 )
706             print( "x166: ", event.x166 )
707             print( "y166: ", event.y166 )
708             print( "z166: ", event.z166 )
709             print( "x167: ", event.x167 )
710             print( "y167: ", event.y167 )
711             print( "z167: ", event.z167 )
712             print( "x168: ", event.x168 )
713             print( "y168: ", event.y168 )
714             print( "z168: ", event.z168 )
715             print( "x169: ", event.x169 )
716             print( "y169: ", event.y169 )
717             print( "z169: ", event.z169 )
718             print( "x170: ", event.x170 )
719             print( "y170: ", event.y170 )
720             print( "z170: ", event.z170 )
721             print( "x171: ", event.x171 )
722             print( "y171: ", event.y171 )
723             print( "z171: ", event.z171 )
724             print( "x172: ", event.x172 )
725             print( "y172: ", event.y172 )
726             print( "z172: ", event.z172 )
727             print( "x173: ", event.x173 )
728             print( "y173: ", event.y173 )
729             print( "z173: ", event.z173 )
730             print( "x174: ", event.x174 )
731             print( "y174: ", event.y174 )
732             print( "z174: ", event.z174 )
733             print( "x175: ", event.x175 )
734             print( "y175: ", event.y175 )
735             print( "z175: ", event.z175 )
736             print( "x176: ", event.x176 )
737             print( "y176: ", event.y176 )
738             print( "z176: ", event.z176 )
739             print( "x177: ", event.x177 )
740             print( "y177: ", event.y177 )
741             print( "z177: ", event.z177 )
742             print( "x178: ", event.x178 )
743             print( "y178: ", event.y178 )
744             print( "z178: ", event.z178 )
745             print( "x179: ", event.x179 )
746             print( "y179: ", event.y179 )
747             print( "z179: ", event.z179 )
748             print( "x180: ", event.x180 )
749             print( "y180: ", event.y180 )
750             print( "z180: ", event.z180 )
751             print( "x181: ", event.x181 )
752             print( "y181: ", event.y181 )
753             print( "z181: ", event.z181 )
754             print( "x182: ", event.x182 )
755             print( "y182: ", event.y182 )
756             print( "z182: ", event.z182 )
757             print( "x183: ", event.x183 )
758             print( "y183: ", event.y183 )
759             print( "z183: ", event.z183 )
760             print( "x184: ", event.x184 )
761             print( "y184: ", event.y184 )
762             print( "z184: ", event.z184 )
763             print( "x185: ", event.x185 )
764             print( "y185: ", event.y185 )
765             print( "z185: ", event.z185 )
766             print( "x186: ", event.x186 )
767             print( "y186: ", event.y186 )
768             print( "z186: ", event.z186 )
769             print( "x187: ", event.x187 )
770             print( "y187: ", event.y187 )
771             print( "z187: ", event.z187 )
772             print( "x188: ", event.x188 )
773             print( "y188: ", event.y188 )
774             print( "z188: ", event.z188 )
775             print( "x189: ", event.x189 )
776             print( "y189: ", event.y189 )
777             print( "z189: ", event.z189 )
778             print( "x190: ", event.x190 )
779             print( "y190: ", event.y190 )
780             print( "z190: ", event.z190 )
781             print( "x191: ", event.x191 )
782             print( "y191: ", event.y191 )
783             print( "z191: ", event.z191 )
784             print( "x192: ", event.x192 )
785             print( "y192: ", event.y192 )
786             print( "z192: ", event.z192 )
787             print( "x193: ", event.x193 )
788             print( "y193: ", event.y193 )
789             print( "z193: ", event.z193 )
790             print( "x194: ", event.x194 )
791             print( "y194: ", event.y194 )
792             print( "z194: ", event.z194 )
793             print( "x195: ", event.x195 )
794             print( "y195: ", event.y195 )
795             print( "z195: ", event.z195 )
796             print( "x196: ", event.x196 )
797             print( "y196: ", event.y196 )
798             print( "z196: ", event.z196 )
799             print( "x197: ", event.x197 )
800             print( "y197: ", event.y197 )
801             print( "z197: ", event.z197 )
802             print( "x198: ", event.x198 )
803             print( "y198: ", event.y198 )
804             print( "z198: ", event.z198 )
805             print( "x199: ", event.x199 )
806             print( "y199: ", event.y199 )
807             print( "z199: ", event.z199 )
808             print( "x200: ", event.x200 )
809             print( "y200: ", event.y200 )
810             print( "z200: ", event.z200 )
811             print( "x201: ", event.x201 )
812             print( "y201: ", event.y201 )
813             print( "z201: ", event.z201 )
814             print( "x202: ", event.x202 )
815             print( "y202: ", event.y202 )
816             print( "z202: ", event.z202 )
817             print( "x203: ", event.x203 )
818             print( "y203: ", event.y203 )
819             print( "z203: ", event.z203 )
820             print( "x204: ", event.x204 )
821             print( "y204: ", event.y204 )
822             print( "z204: ", event.z204 )
823             print( "x205
```

```

200     print( "markerId: ", event.markerId ) -- ID of the marker that was touched
201     print( "lat: ", event.latitude ) -- latitude of the marker
202     print( "long: ", event.longitude ) -- longitude of the marker
203
204     end
205
206     myMap:setCenter( currentLocation.latitude, currentLocation.longitude )
207
208
209     end
210 end
211 locationHandler()
212
213
214 function openMenu()
215
216     --myMap.x = 270
217     --myMap.width = myMap.width-270
218     local currScene = composer.getSceneName( "current" )
219     if currScene then
220         composer.removeScene( currScene )
221     end
222     composer.gotoScene("menu",{effect = "slideRight",time = 400})
223
224 end
225
226
227 button:addEventListener("tap",openMenu)
228
229 -----
230 -- botao de busca -----
231
232 campodebusca = native.newTextField(350, 35, 300, 50)
233
234 local function BuscarEndereco( event )
235     if ( "ended" == event.phase ) then
236         native.setKeyboardFocus(nil)
237         if campodebusca.text ~= nil and campodebusca.text ~= "" then
238             local function locationHandler( event )
239                 local data = json.decode(event.response)
240                 if ( event.isError ) then
241                     print( "Network error: ", event.response )
242                     native.showAlert("Atenção","Verifique sua internet e tente novamente.",{"OK"})
243                 else
244                     if data and data.status then
245                         if data.status == "ZERO_RESULTS" then
246                             native.showAlert("Não encontrado","Não foi possível encontrar o endereço!\n"..campodebusca.text,{"OK"})
247                         elseif data.status == "OVER_DAILY_LIMIT" then
248                             native.showAlert("Passou o limite Diário","API key inválida, ou cobrança não configurada ou inválida para a conta, ou excedeu o limite de acessos.",{"OK"})
249                         elseif data.status == "Passou o limite de tráfego" then
250                             native.showAlert(data.status,"Excedeu sua quota de acessos",{"OK"})
251                         elseif data.status == "REQUEST_DENIED" then
252                             native.showAlert("Requerimento Negado","verifique sua internet e tente novamente",{"OK"})
253                         elseif data.status == "INVALID_REQUEST" then
254                             native.showAlert("Requisição Inválida",{"OK"})
255                     end
256                 end
257             end
258             locationHandler()
259         end
260     end
261 end

```

```

253 elseif data.status == "Requerimento Inválido" then
254     native.showAlert(data.status,"Falta algum componente de endereço para completar a busca.",{"OK"})
255 elseif data.status == "Erro desconhecido" then
256     native.showAlert(data.status,"Não pode prosseguir por um erro no servidor",{"OK"})
257 elseif data.status == "OK" then
258     local resultados = data.results
259
260     local geometria = resultados[1].geometry
261     local localizacao = geometria.location
262     local latitude = localizacao.lat
263     local longitude = localizacao.lng
264     if markerID then
265         if myMap then
266             myMap:removeMarker( markerID )
267         end
268         markerID = nil
269     end
270
271     if myMap then
272         myMap:setCenter( latitude, longitude )
273         markerID =myMap:addMarker(latitude,longitude)
274     end
275 end
276 else
277     native.showAlert("Endereço Incompleto","Complemente o campo de busca e tente novamente.",{"try again"})
278 end
279 end
280 end
281 network.request("https://maps.googleapis.com/maps/api/geocode/json?address=..campodebusca.text..&key=AIzaSyBrAkaiJUDdUvS-idGVphnWnxnFrJvjTjs","GET",locationHandler)
282 end
283 end
284 end
285
286 botaoBusca = widget.newButton(
287 {
288     width = 163,
289     height = 54,
290     defaultFile = "buscaUp.png",
291     overFile = "buscaDown.png",
292     onEvent = BuscarEndereco
293 }
294 )
295 botaoBusca.anchorX = 1
296 botaoBusca.anchorY = 0
297 botaoBusca.x = W - 10
298 botaoBusca.y = 10
299
300 -----
301 -----
302 -- PEGAR BASE DE DADOS -----
303 local cadastros = {}
304 local cartoes = {}

```

```

305 local enderecos = {}
306 local URL = "http://www.coronasdkgames.com/Gabriel/importarBD.php"
307 parametro = {}
308 parametro.body = "tipo=cadastros"
309
310 local function CriarBasesDeDados()
311     if cadastros[1] then
312         cadastros.id = {}
313         cadastros.login = {}
314         cadastros.senha = {}
315         cadastros.email = {}
316         cadastros.nome = {}
317         cadastros.CPF = {}
318         for i=1,#cadastros do
319             cadastros.id[i] = cadastros[i][1]
320             cadastros.login[i] = cadastros[i][2]
321             cadastros.senha[i] = cadastros[i][3]
322             cadastros.email[i] = cadastros[i][4]
323             cadastros.nome[i] = cadastros[i][5]
324             cadastros.CPF[i] = cadastros[i][6]
325         end
326         --print(cadastros.id[1],cadastros.login[1],cadastros.senha[1],cadastros.email[1],cadastros.nome[1],cadastros.CPF[1])
327         local path = system.pathForFile("cadastros.sqlite", system.DocumentsDirectory)
328         print(path)
329         db = sqlite3.open (path)
330
331         local tablesetup = "CREATE TABLE IF NOT EXISTS cadastros (id PRIMARY KEY, Nomecompleto, Login, Senha, Email, CPF);"
332         db:exec(tablesetup)
333         print (tablesetup)
334         for i=1,#cadastros do
335             local tablefill = "INSERT INTO cadastros VALUES('.. cadastros.id[i] ..','.. cadastros.nome[i] ..','.. cadastros.login[i] ..',
336                 '.. cadastros.senha[i] ..','.. cadastros.email[i] ..','.. cadastros.CPF[i] ..');"
337             print (tablefill)
338             db:exec(tablefill)
339         end
340         db:close()
341         print("db closed")
342     end
343     if cartoes[1] then
344         cartoes.id = {}
345         cartoes.CPF = {}
346         cartoes.numero = {}
347         cartoes.codigo = {}
348         cartoes.nome = {}
349         cartoes.validade = {}
350         for i=1,#cartoes do
351             cartoes.id[i] = cartoes[i][1]
352             cartoes.CPF[i] = cartoes[i][2]
353             cartoes.numero[i] = cartoes[i][3]
354             cartoes.codigo[i] = cartoes[i][4]
355             cartoes.nome[i] = cartoes[i][5]
356             cartoes.validade[i] = cartoes[i][6]

```

```

357     end
358
359
360     local path = system.pathForFile("cartoes.sqlite", system.DocumentsDirectory)
361     print(path)
362     db = sqlite3.open (path)
363
364     local tablesetup = "CREATE TABLE IF NOT EXISTS cartoes (id INTEGER PRIMARY KEY, CPF, NdCartao, CodigodoCartao, DatadeVencimento);"
365     db:exec(tablesetup)
366     print (tablesetup)
367     for i=1,#cartoes do
368         local tablefill = "INSERT INTO cartoes VALUES('.. cartoes.id[i] ..','.. cartoes.CPF[i] ..','.. cartoes.numero[i] ..',
369             '.. cartoes.codigo[i] ..','.. cartoes.nome[i] ..','.. cartoes.validade[i] ..');"
370         print (tablefill)
371         db:exec(tablefill)
372     end
373     db:close()
374     print("db closed")
375 end
376 if enderecos[1] then
377     enderecos.id = {}
378     enderecos.endereco = {}
379     enderecos.latitude = {}
380     enderecos.longitude = {}
381     enderecos.CPF = {}
382     enderecos.pequenos = {}
383     enderecos.medios = {}
384     enderecos.grandes = {}
385     enderecos.motos = {}
386     enderecos.pequenos0 = {}
387     enderecos.medios0 = {}
388     enderecos.grandes0 = {}
389     enderecos.motos0 = {}
390     enderecos.nome = {}
391     for i=1,#enderecos do
392         enderecos.id[i] = enderecos[i][1]
393         enderecos.endereco[i] = enderecos[i][2]
394         enderecos.latitude[i] = enderecos[i][3]
395         enderecos.longitude[i] = enderecos[i][4]
396         enderecos.CPF[i] = enderecos[i][5]
397         enderecos.pequenos[i] = enderecos[i][6]
398         enderecos.medios[i] = enderecos[i][7]
399         enderecos.grandes[i] = enderecos[i][8]
400         enderecos.motos[i] = enderecos[i][9]
401         enderecos.pequenos0[i] = enderecos[i][10]
402         enderecos.medios0[i] = enderecos[i][11]
403         enderecos.grandes0[i] = enderecos[i][12]

```

```

404     enderecos.motos0[i] = enderecos[i][13]
405     enderecos.nome[i] = enderecos[i][14]
406 end
407 --print(enderecos.id[1],enderecos.endereco[1],enderecos.latitude[1],enderecos.longitude[1],enderecos.CPF[1])
408
409 local path = system.pathForFile("garagens.sqlite", system.DocumentsDirectory)
410 print(path)
411 db = sqlite3.open (path)
412
413 local tablesetup = "CREATE TABLE IF NOT EXISTS enderecos (id INTEGER PRIMARY KEY, endereco, latitude, longitude,CPF,pequenos,medios,grandes,motos,pequenos0,medios0,grandes0,motos0,nome);"
414 db:exec(tablesetup)
415 print (tablesetup)
416 for i=1,#enderecos do
417     local tablefill = "INSERT INTO enderecos VALUES('".. enderecos.id[i] .. "','".. enderecos.endereco[i] .. "','".. enderecos.latitude[i] .. "','".. enderecos.longitude[i] .. "',
418     '".. enderecos.CPF[i] .. "','".. enderecos.pequenos[i] .. "','".. enderecos.medios[i] .. "','".. enderecos.grandes[i] .. "','".. enderecos.motos[i] .. "',
419     '".. enderecos.pequenos0[i] .. "','".. enderecos.medios0[i] .. "','".. enderecos.grandes0[i] .. "','".. enderecos.motos0[i] .. "','".. enderecos.nome[i] ..');"
420     print (tablefill)
421     db:exec(tablefill)
422 end
423 db:close()
424 print("db closed")
425 -- CRIAR MARCADORES --
426 local path = system.pathForFile("garagens.sqlite", system.DocumentsDirectory)
427 print(path)
428 db = sqlite3.open (path)
429
430 for row in db:nrows("SELECT * FROM enderecos ") do
431     local labels = {}
432
433     local function markerListener(event)
434         longitudedagaragem = event.longitude
435         latitudedagaragem = event.latitude
436         iddagaragem = row.id
437     end
438     local vetor = {row.pequenos - row.medios}
439     local options =
440     {
441         title = row.nome,
442         subtitle = row.endereco.."\\n"..VAGAS\\nPequenas "..row.pequenos.."\\nMedias "..row.medios.."\\nGrandes "..row.grandes.."\\nMotos "..row.motos,
443         imageFile = "marker.png",
444         listener = markerListener,
445     }
446
447     myMap:addMarker(tonumber(row.latitude),tonumber(row.longitude),options)
448 end
449 --
450 end
451 end
452 local function networkListener4( event )
453
454 if ( event.isError ) then
455     local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )

```

```

456     print(event.response)
457 else
458
459     if event.response ~= "null " then
460         enderecos = json.decode(event.response)
461     end
462
463     CriarBasesDeDados()
464 end
465 end
466 local function networkListener3( event )
467
468     if ( event.isError ) then
469         local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )
470         print(event.response)
471     else
472
473         if event.response ~= "null " then
474             cartoes = json.decode(event.response)
475         end
476         --local alert = native.showAlert( "Conectando à base de dados online", "Base de daods lida com sucesso\n"..event.response, { "Ok" } )
477         parametro.body = "tipo=enderecos"
478         network.request(URL, "POST", networkListener4,parametro) -- baixando base de dados completa
479     end
480 end
481 local function networkListener2( event )
482     print(event.response)
483     if ( event.isError ) then
484         local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )
485     else
486
487
488         if event.response ~= "null " then
489             cadastros = json.decode(event.response)
490         end
491         --local alert = native.showAlert( "Conectando à base de dados online", "Base de daods lida com sucesso\n"..event.response, { "Ok" } )
492
493         parametro.body = "tipo=cartoes"
494         network.request(URL, "POST", networkListener3,parametro) -- baixando base de dados completa
495     end
496 end
497
498
499 network.request(URL, "POST", networkListener2,parametro) -- baixando base de dados completa
500

```

Menu.lua.

```

1  local composer = require("composer")
2  local scene = composer.newScene()
3  local localGroup = display.newGroup()
4  local widget = require ("widget")
5  local db
6  local sqlite3 = require("sqlite3")
7
8  local latitude
9  local longitude
10
11 function scene:create (event)
12
13     campodebusca.isVisible = true
14     columnData.isVisible = true
15     botaoReserva.isVisible = true
16     botaoBusca.isVisible = true
17     Interface.isVisible = true
18     pickerWheel.isVisible = true
19     myMap.isVisible = true
20
21     myMap.x=270
22     myMap.width = W-270
23     campodebusca.width =250
24     campodebusca.x = 400
25
26     if ( LoginNumber == 0 ) then
27         botaoData.isVisible = false
28     else
29         botaoData.isVisible = true
30     end
31
32     local sceneGroup = self.view
33
34     local cadastrar_function = function (event)
35         composer.removeScene( "menu" )
36         composer.gotoScene("cadastrar", "slideRight", 400)
37     end
38
39     local Login_function = function (event)
40         composer.removeScene( "menu" )
41         composer.gotoScene("login", "slideRight", 400)
42     end
43
44     local MetododePagamento_function = function (event)
45         composer.removeScene( "menu" )
46         composer.gotoScene("MetododePagamento", "slideRight", 400)
47     end
48
49     local AddGaragem_function = function (event)
50         composer.removeScene( "menu" )
51         composer.gotoScene("AdicionarGaragem", "slideRight", 400)
52     end

```

```

54 local SelecionData_function = function (event)
55     composer.removeScene( "menu" )
56     composer.gotoScene("Data", "slideRight", 400)
57 end
58
59
60
61 function removeMenuClicandoFora(menu,login,cadastrar)
62
63     local cliqueFora = display.newRect(0,0,0,0)
64     cliqueFora.anchorX=0
65     cliqueFora.anchorY=0
66     cliqueFora.width = W
67     cliqueFora.height = H
68     cliqueFora.x = 0
69     cliqueFora.y = 0
70     cliqueFora.alpha = 0
71     cliqueFora.isHitTestable = true
72     menu:addEventListener("tap",function() return true; end)
73     localGroup:insert(cliqueFora)
74     localGroup:insert(menu)
75     localGroup:insert(login)
76     localGroup:insert(cadastrar)
77 local function removeMenuAux()
78     myMap.x = 0
79     myMap.width = W
80     campodebusca.width =300
81     campodebusca.x = 350
82     cadastrar:removeSelf()
83     login:removeSelf()
84     menu:removeSelf()
85     cliqueFora:removeEventListener("tap",removeMenuAux)
86     cliqueFora:removeSelf()
87 end
88 cliqueFora:addEventListener("tap",removeMenuAux)
89 end
90
91
92
93 function removeMenuClicandoFora2(menu,MetododePagamento,AddGaragem,SelectData)
94
95     local cliqueFora = display.newRect(0,0,0,0)
96     cliqueFora.anchorX=0
97     cliqueFora.anchorY=0
98     cliqueFora.width = W
99     cliqueFora.height = H
100    cliqueFora.x = 0
101    cliqueFora.y = 0
102    cliqueFora.alpha = 0
103    cliqueFora.isHitTestable = true
104    menu:addEventListener("tap",function() return true; end)
105    localGroup:insert(cliqueFora)

```

```

106     localGroup:insert(menu)
107     localGroup:insert(AddGaragem)
108     localGroup:insert(MetododePagamento)
109     localGroup:insert(SelectData)
110     local function removeMenuAux()
111         myMap.x = 0
112         myMap.width = W
113         campodebusca.width = 300
114         campodebusca.x = 350
115         AddGaragem:removeSelf()
116         MetododePagamento:removeSelf()
117         menu:removeSelf()
118         SelectData:removeSelf()
119         cliqueFora:removeEventListener("tap", removeMenuAux)
120         cliqueFora:removeSelf()
121     end
122     cliqueFora:addEventListener("tap", removeMenuAux)
123 end
124
125 local menu = display.newImage("MenuAberto.png", W/4, 4*H/5)
126 menu.anchorX = 0
127 menu.anchorY = 0
128 menu.x=0
129 menu.y=0
130
131 if ( LoginNumber == 0 ) then
132     local login = widget.newButton{
133         defaultFile = "Button.png",
134         overFile = "Button.png",
135         label = "Login",
136         fontSize = 44,
137         labelColor = {default = {1,1,1}},
138         emboss=true,
139         onRelease = Login_function,
140         x = 120,
141         y = 350,
142     }
143     local cadastrar = widget.newButton{
144         defaultFile = "Button.png",
145         overFile = "Button.png",
146         label = "Cadastrar",
147         fontSize = 44,
148         labelColor = {default = {1,1,1}},
149         emboss=true,
150         onRelease = cadastrar_function,
151         x = 120,
152         y = 250,
153     }
154     menu.anchorX = 0
155     menu.anchorY = 0
156     menu.x=0
157     menu.y=0

```

```

159 removeMenuClicandoFora(menu,login,cadastrar)
160 else
161     local MetododePagamento = widget.newButton{
162         defaultFile = "Button.png",
163         overFile = "Button.png",
164         label = "Cartões",
165         fontSize = 28,
166         labelColor = {default = {1,1,1}},
167         emboss=true,
168         onRelease = MetododePagamento_function,
169         x = 120,
170         y = 350,
171     }
172
173
174     local AddGaragem = widget.newButton{
175         defaultFile = "Button.png",
176         overFile = "Button.png",
177         label = "Adicionar Garagem",
178         fontSize = 28,
179         labelColor = {default = {1,1,1}},
180         emboss=true,
181         onRelease = AddGaragem_function,
182         x = 120,
183         y = 250,
184     }
185
186     local SelectData = widget.newButton{
187         defaultFile = "Button.png",
188         overFile = "Button.png",
189         label = "Escolha uma data",
190         fontSize = 28,
191         labelColor = {default = {1,1,1}},
192         emboss=true,
193         onRelease = SelecioneData_function,
194         x = 120,
195         y = 450,
196     }
197
198     removeMenuClicandoFora2(menu,MetododePagamento,AddGaragem, SelectData)
199 end
200
201 end
202 function scene:destroy( event )
203
204     campodebusca.isVisible = false
205     myMap.isVisible = false
206     columnData.isVisible = false
207     botaoReserva.isVisible = false
208     botaoBusca.isVisible = false
209     Interface.isVisible = false
210     pickerWheel.isVisible = false

```

```

211     botaoData.isVisible = false
212     local sceneGroup = self.view
213     localGroup:removeSelf()
214
215 end
216
217 scene:addEventListener( "create", scene )
218 scene:addEventListener("destroy", scene)
219
220 return scene

```

Cadastro.lua

```

1  local composer = require("composer")
2  local scene = composer.newScene()
3  local localGroup = display.newGroup()
4  local widget = require("widget")
5  local db
6  local cpfLib = require("cpfLib")
7
8  local sqlite3 = require("sqlite3")
9
10 local function networkListener( event )
11     if ( event.isError ) then
12         print( "Network error!" )
13     else
14         print ( "RESPONSE: " .. event.response )
15     end
16 end
17
18 function scene:create (event)
19     local title = display.newText(localGroup, "Ficha de Cadastro", 350, 50, nil, 48)
20     title:setTextColor(1,1,1)
21     local nameLabel = display.newText(localGroup, "Nome Completo", 150, 200, nil, 24)
22     nameLabel:setTextColor(1,1,1)
23     local LoginLabel = display.newText(localGroup, "Login", 150, 300, nil, 24)
24     LoginLabel:setTextColor(1,1,1)
25     local senhaLabel = display.newText(localGroup, "Senha", 150, 400, nil, 24)
26     senhaLabel:setTextColor(1,1,1)
27     local confirmarsenhaLabel = display.newText(localGroup, "Confirmar Senha", 150, 500, nil, 24)
28     confirmarsenhaLabel:setTextColor(1,1,1)
29     local emailLabel = display.newText(localGroup, "e-mail", 150, 600, nil, 24)
30     emailLabel:setTextColor(1,1,1)
31     local CPFLabel = display.newText(localGroup, "CPF", 150, 700, nil, 24)
32     CPFLabel:setTextColor(1,1,1)
33
34
35
36     local NomedoCliente = native.newTextField(400, 200, 220, 36)
37     localGroup:insert(NomedoCliente)
38     local LogindoCliente = native.newTextField(400, 300, 220, 36)
39     localGroup:insert(LogindoCliente)
40     local SenhadoCliente = native.newTextField(400, 400, 220, 36)
41     localGroup:insert(SenhadoCliente)
42     SenhadoCliente.isSecure = true
43     local Senha2 = native.newTextField(400, 500, 220, 36)
44     localGroup:insert(Senha2)
45     Senha2.isSecure = true
46     local EmaildoCliente = native.newTextField(400, 600, 220, 36)
47     localGroup:insert(EmaildoCliente)
48     local CPFdoCliente = native.newTextField(400, 700, 220, 36)
49     localGroup:insert(CPFdoCliente)
50     CPFdoCliente.inputType = "number"
51
52     local submitCadastro_function = function (event)

```

```

53 local resultado = cpfLib.verificarCPF(CPFdoCliente.text)
54 if ( resultado == true) then
55     if (SenhaCliente.text == Senha2.text) then
56         local path = system.pathForFile("cadastros.sqlite", system.DocumentsDirectory)
57         print(path)
58         db = sqlite3.open (path)
59
60         local tablesetup = "CREATE TABLE IF NOT EXISTS cadastros (ID INTEGER PRIMARY KEY autoincrement, NomeCompleto, Login, Senha, Email, CPF);"
61         db:exec(tablesetup)
62         print (tablesetup)
63         local tablefill = "INSERT INTO cadastros VALUES('.. NomeCliente.text ..', '.. LogindoCliente.text ..',
64             '.. SenhadoCliente.text ..', '.. EmaildoCliente.text ..', '.. CPFdoCliente.text ..');"
65         print (tablefill)
66         db:exec(tablefill)
67         db:close()
68         print("db closed")
69
70 local function cadastralistener( event )
71
72     if ( event.isError ) then
73         local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )
74         print(event.response)
75
76     else
77         if event.response == "Registrado com sucesso ." then
78             local alert = native.showAlert( "Cadastrando no banco de dados", "Resultado: "..event.response, { "Ok" } )
79             NomeCliente:removeSelf()
80             LogindoCliente:removeSelf()
81             SenhadoCliente:removeSelf()
82             EmaildoCliente:removeSelf()
83             --CPFdoCliente:removeSelf()
84
85             composer.removeScene( "cadastrar" )
86             composer.gotoScene("menu", {effect="slideLeft", time = 400})
87         else
88             local alert = native.showAlert( "Erro ao Cadastrar", event.response, { "Try again" } )
89         end
90     end
91 end
92
93 parameters = {}
94 parameters.body = "Register=1&nome=" .. NomeCliente.text .. "&login=" .. LogindoCliente.text .. "&senha=" .. SenhadoCliente.text .. "&email=" .. EmaildoCliente.text .. "&CPF=" .. CPFdoCliente.text
95 local URL2 = "http://www.coronasdkgames.com/Gabriel/cadastrar.php"
96 network.request(URL2, "POST", cadastralistener,parameters) -- baixando base de dados completa
97 else
98     local erro = display.newText(localGroup, "As senhas não são iguais.", 350, 150, nil, 36)
99     erro:setTextColor(1,0,0)
100     localGroup:insert(erro)
101 end
102 else
103     local erro = display.newText(localGroup, "O CPF é Inválido.", 350, 150, nil, 36)
104     erro:setTextColor(1,0,0)

```

```

105         localGroup:insert(erro)
106     end
107
108
109 end
110
111 local addButton_Cadastro = widget.newButton{
112     defaultFile = "Button.png",
113     overFile = "Button.png",
114     label = "Cadastrar",
115     fontSize = 44,
116     labelColor = {default = {1,1,1}},
117     emboss=true,
118     onRelease = submitCadastro_function,
119     x = display.contentWidth/2+100,
120     y = display.contentHeight - 200,
121 }
122 localGroup:insert(addButton_Cadastro)
123
124 local function onSystemEvent (event)
125     if (db and event.type == "applicationExit") then
126         db:close()
127     end
128 end
129
130 Runtime:addEventListener ("system", onSystemEvent)
131 end
132
133 function scene:destroy( event )
134
135     local sceneGroup = self.view
136     localGroup:removeSelf()
137
138 end
139
140 scene:addEventListener( "create", scene )
141 scene:addEventListener("destroy", scene)
142
143 return scene

```

Login.lua

```

1 local composer = require("composer")
2 local scene = composer.newScene()
3 local localGroup = display.newGroup()
4 local widget = require ("widget")
5 local db
6
7 local sqlite3 = require("sqlite3")
8
9 function scene:create (event)
10
11     local title = display.newText(localGroup, "Member Login", 350, 50, nil, 48)
12     title:setTextColor(1,1,1)
13     local LoginLabel = display.newText(localGroup, "Login", 150, 300, nil, 24)
14     LoginLabel:setTextColor(1,1,1)
15     local senhaLabel = display.newText(localGroup, "Senha", 150, 400, nil, 24)
16
17     local LogindoCliente = native.newTextField(400, 300, 220, 36)
18     localGroup:insert(LogindoCliente)
19     local SenhadoCliente = native.newTextField(400, 400, 220, 36)
20     localGroup:insert(SenhadoCliente)
21     SenhadoCliente.isSecure = true
22
23     local submitLogin_function = function (event)
24         local path = system.pathForFile("cadastros.sqlite", system.DocumentsDirectory)
25         print(path)
26         db = sqlite3.open (path)
27
28         for row in db:nrows("SELECT * FROM cadastros") do
29             if(row.Login == LogindoCliente.text) then
30                 if(row.Senha == SenhadoCliente.text) then
31                     --erro:removeSelf()
32                     LoginNumber=1
33                     CPF = row.CPF
34                     print(CPF)
35                     db:close()
36                     composer.removeScene( "login" )
37                     composer.gotoScene("menu", {effect="slideLeft", time = 400})
38
39                     return true
40                 else
41                     erro = display.newText(localGroup, "O Login ou senha está incorreto", 350, 150, nil, 36)
42                     erro:setTextColor(1,0,0)
43                     localGroup:insert(erro)
44
45                     end
46                 end
47             else
48                 erro = display.newText(localGroup, "O Login ou senha está incorreto", 350, 150, nil, 36)
49                 erro:setTextColor(1,0,0)
50                 localGroup:insert(erro)
51             end
52         end
53     end
54 end

```

```
53 |         end
54 |     end
55 |
56 |     local addButton_Login = widget.newButton{
57 |         defaultFile = "Button.png",
58 |         overFile = "Button.png",
59 |         label = "Entrar",
60 |         fontSize = 44,
61 |         labelColor = {default = {1,1,1}},
62 |         emboss=true,
63 |         onRelease = submitLogin_function,
64 |         x = display.contentWidth/2+100,
65 |         y = display.contentHeight - 200,
66 |     }
67 |     localGroup:insert(addButton_Login)
68 |
69 |
70 |
71 | end
72 |
73 | function scene:destroy( event )
74 |     local sceneGroup = self.view
75 |     localGroup:removeSelf()
76 |
77 | end
78 |
79 | scene:addEventListener( "create", scene )
80 | scene:addEventListener("destroy", scene)
81 |
82 | return scene
```

MetododePagamento.lua

```

1  local composer = require("composer")
2  local scene = composer.newScene()
3  local localGroup = display.newGroup()
4  local widget = require("widget")
5  local db
6
7  local sqlite3 = require("sqlite3")
8
9  function scene:create (event)
10
11     local voltar_function = function (event)
12         composer.removeScene( "MetododePagamento" )
13         composer.gotoScene("menu", "slideLeft", 400)
14     end
15
16     local CadastrodeCartao_function = function (event)
17         composer.removeScene( "MetododePagamento" )
18         composer.gotoScene("CadastrodeCartao", "slideRight", 400)
19     end
20
21     local AdicionarCartao = widget.newButton{
22         defaultFile = "Button.png",
23         overFile = "Button.png",
24         label = "Adicionar Cartão",
25         fontSize = 44,
26         labelColor = {default = {1,1,1}},
27         emboss=true,
28         onRelease = CadastrodeCartao_function,
29         x = display.contentWidth/2,
30         y = 450,
31     }
32     localGroup:insert(AdicionarCartao)
33
34     local Voltar = widget.newButton{
35         defaultFile = "Button.png",
36         overFile = "Button.png",
37         label = "Voltar",
38         fontSize = 44,
39         labelColor = {default = {1,1,1}},
40         emboss=true,
41         onRelease = voltar_function,
42         x = display.contentWidth/2,
43         y = 650,
44     }
45     localGroup:insert(Voltar)
46
47 end
48
49 function scene:destroy( event )
50
51     local sceneGroup = self.view
52     localGroup:removeSelf()

```

```

54 end
55
56 scene:addEventListener( "create", scene )
57 scene:addEventListener("destroy", scene)
58
59 return scene

```

CadastrodeCartao.lua

```

1 local composer = require("composer")
2 local scene = composer.newScene()
3 local localGroup = display.newGroup()
4 local widget = require("widget")
5 local db
6 local cpfLib = require("cpfLib")
7
8 local sqlite3 = require("sqlite3")
9
10 function scene:create (event)
11
12     local title = display.newText(localGroup, "Cadastrar um novo cartão de crédito", 350, 150, nil, 36)
13     title:setTextColor(1,1,1)
14     local cpfLabel = display.newText(localGroup, "CPF do titular", 150, 300, nil, 24)
15     cpfLabel:setTextColor(1,1,1)
16     local ndcartaoLabel = display.newText(localGroup, "número do cartão", 130, 400, nil, 24)
17     ndcartaoLabel:setTextColor(1,1,1)
18     local codigoLabel = display.newText(localGroup, "Código de segurança", 120, 500, nil, 24)
19     codigoLabel:setTextColor(1,1,1)
20     local ddvLabel = display.newText(localGroup, "Data de Validade", 130, 600, nil, 24)
21     ddvLabel:setTextColor(1,1,1)
22     local nomeLabel = display.newText(localGroup, "Nome no cartão", 130, 700, nil, 24)
23     nomeLabel:setTextColor(1,1,1)
24
25
26     local CpfCliente = native.newTextField(400, 300, 320, 36)
27     localGroup:insert(CpfCliente)
28     CpfCliente.inputType = "number"
29     local NdcartaoCliente = native.newTextField(400, 400, 320, 36)
30     localGroup:insert(NdcartaoCliente)
31     local codigoCliente = native.newTextField(400, 500, 320, 36)
32     localGroup:insert(codigoCliente)
33     local ddvCliente = native.newTextField(400, 600, 320, 36)
34     localGroup:insert(ddvCliente)
35     local nomeCliente = native.newTextField(400, 700, 320, 36)
36     localGroup:insert(nomeCliente)
37
38     local submitCadastro_function = function (event)
39         local resultado = cpfLib. verificarCPF(CpfCliente.text)
40         if (resultado == true) then
41             local path = system.pathForFile("cartoes.sqlite", system.DocumentsDirectory)
42             print(path)
43             db = sqlite3.open (path)
44
45             local tablesetup = "CREATE TABLE IF NOT EXISTS cartoes (ID INTEGER PRIMARY KEY, CPF, NdCartao, CodigodoCartao, DatadeVencimento, Nomenocartao);"
46             db:exec(tablesetup)
47             print (tablesetup)
48             local tablefill = "INSERT INTO cartoes VALUES('.. CpfCliente.text ..', '.. NdcartaoCliente.text ..', '.. codigoCliente.text ..',
49                 '.. nomeCliente.text ..', '.. ddvCliente.text ..', '..nomeCliente.text..');"
50             print (tablefill)
51             db:exec(tablefill)
52             db:close()

```

```

53     print("db closed")
54
55     local function cadastralListener( event )
56
57         if ( event.isError ) then
58             local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )
59             print(event.response)
60
61         else
62             if event.response == "Registrado com sucesso ." then
63                 local alert = native.showAlert( "Cadastrando no banco de dados", "Resultado: "..event.response, { "Ok" } )
64                 CpfCliente:removeSelf()
65                 NdcartaoCliente:removeSelf()
66                 codigoCliente:removeSelf()
67                 ddvCliente:removeSelf()
68                 nomeCliente:removeSelf()
69
70                 composer.removeScene( "CadastradoCartao" )
71                 composer.gotoScene("menu", {effect="slideLeft", time = 400})
72             else
73                 local alert = native.showAlert( "Erro ao Cadastrar", event.response, { "Try again" } )
74             end
75         end
76     end
77
78     parameters = {}
79     parameters.body = "Register=1&CPF=" .. CpfCliente.text .. "&numero=" .. NdcartaoCliente.txt .. "&codigo=" .. codigoCliente.text .. "&nome=" .. nomeCliente.text .. "&$validade=" .. ddvCliente.text
80     local URL2 = "http://www.coronasdkgames.com/Gabriel/cadastrarCartao.php"
81     network.request(URL2, "POST", cadastralListener,parameters) -- baixando base de dados completa
82
83     else
84         erro = display.newText(localGroup, "O CPF é Inválido.", 350, 050, nil, 36)
85         erro:setTextColor(1,0,0)
86         localGroup:insert(erro)
87     end
88
89
90     local Voltar function = function (event)
91         composer.removeScene( "CadastradoCartao" )
92         composer.gotoScene("MetododePagamento", "slideLeft", 400)
93     end
94
95     local addButton_Cadastro = widget.newButton{
96         defaultFile = "Button.png",
97         overFile = "Button.png",
98         label = "Cadastrar",
99         fontSize = 44,
100         labelColor = {default = {1,1,1}},
101         emboss=true,
102         onRelease = submitCadastro function,
103         x = display.contentWidth/2+150,
104         y = display.contentHeight - 200,

```

```

105         localGroup:insert(erro)
106     end
107
108
109 end
110
111 local addButton_Cadastro = widget.newButton{
112     defaultFile = "Button.png",
113     overFile = "Button.png",
114     label = "Cadastrar",
115     fontSize = 44,
116     labelColor = {default = {1,1,1}},
117     emboss=true,
118     onRelease = submitCadastro_function,
119     x = display.contentWidth/2+100,
120     y = display.contentHeight - 200,
121 }
122 localGroup:insert(addButton_Cadastro)
123
124 local function onSystemEvent (event)
125     if (db and event.type == "applicationExit") then
126         db:close()
127     end
128 end
129
130 Runtime:addEventListener ("system", onSystemEvent)
131 end
132
133 function scene:destroy( event )
134
135     local sceneGroup = self.view
136     localGroup:removeSelf()
137
138 end
139
140 scene:addEventListener( "create", scene )
141 scene:addEventListener("destroy", scene)
142
143 return scene

```

AdicionarGaragem.lua

```

1  local composer = require("composer")
2  local scene = composer.newScene()
3  local localGroup = display.newGroup()
4  local widget = require("widget")
5  local json = require("json")
6  local db
7
8  local sqlite3 = require("sqlite3")
9
10 CPF = "097" -- CPF GENERICO
11 local markerID
12
13 function scene:create (event)
14     local sceneGroup = self.view
15     local title = display.newText(localGroup, "ADICIONAR UMA NOVA GARAGEM", 350, 150, nil, 36)
16     title:setTextColor(1,1,1)
17     local addButton Cadastro
18     local nomeLabel = display.newText(localGroup, "NOME da garagem: ", 0, 200, nil, 24)
19     nomeLabel.anchorX = 0
20     nomeLabel:setTextColor(1,1,1)
21     localGroup:insert(nomeLabel)
22
23     local NomedaGaragem = native.newTextField(100, nomeLabel.y, 320, 36)
24     NomedaGaragem.x = NomedaGaragem.width/2 + nomeLabel.x + nomeLabel.width + 20
25     localGroup:insert(NomedaGaragem)
26
27     local VAGASLabel = display.newText(localGroup, "TIPOS DE VAGAS (Quantidade)\n----- ", 0, 280, nil, 30)
28     VAGASLabel.anchorX = 0
29     VAGASLabel:setTextColor(1,1,1)
30     localGroup:insert(VAGASLabel)
31     local vetorVagas = {"pequena", "media", "grande", "moto"}
32     local vetorLabels = {}
33     local vetorCampos = {}
34     for i=1,#vetorVagas do
35         vetorLabels[i] = display.newText(localGroup, vetorVagas[i]..": ", 0, 300 + 50*i, nil, 24)
36         vetorLabels[i].anchorX = 0
37         vetorLabels[i]:setTextColor(1,1,1)
38
39         vetorCampos[i] = native.newTextField(100, vetorLabels[i].y, 70, 36)
40         vetorCampos[i].x = vetorCampos[i].width/2 + vetorLabels[i].x + vetorLabels[i].width + 20
41         vetorCampos[i].text = 0
42         vetorCampos[i].inputType = "number"
43
44         localGroup:insert(vetorCampos[i])
45     end
46     local VAGASLabel2 = display.newText(localGroup, "----- ", 0, 550, nil, 30)
47     VAGASLabel2.anchorX = 0
48     VAGASLabel2:setTextColor(1,1,1)
49
50     local function textListener( event )
51         if ( event.phase == "editing" ) then
52             addButton Cadastro.isVisible = false

```



```

105     if markerID then
106         if myMap2 then
107             myMap2:removeMarker( markerID )
108         end
109         markerID = nil
110     end
111
112     if myMap2 then
113         myMap2:setCenter( latitude, longitude )
114         markerID = myMap2:addMarker(latitude,longitude)
115     end
116     addButton_Cadastro.isVisible = true
117 end
118 else
119     native.showAlert("Endereço Incompleto","Complemente o campo de busca e tente novamente.",{"try again"})
120 end
121 end
122 end
123     network.request("https://maps.googleapis.com/maps/api/geocode/json?address="..EnderecoCliente.text.."&key=AlzaSyBrAkaiJUDdÜvS-idGVphnWnxnFrJvjTjs","GET",locationHandler)
124 end
125 end
126 end
127 local botaoBusca2 = widget.newButton(
128 {
129     width = 163,
130     height = 54,
131     defaultFile = "buscaUp.png",
132     overFile = "buscaDown.png",
133     onEvent = BuscarEndereco
134 }
135 )
136 botaoBusca2.anchorX = 0
137 botaoBusca2.anchorY = .5
138 botaoBusca2.x = EnderecoCliente.x + EnderecoCliente.width/2
139 botaoBusca2.y = EnderecoCliente.y
140 localGroup:insert(botaoBusca2)
141 --latitudeText,longitudeText = myMap2:getUserLocation()
142 local submitCadastro_function = function (event)
143     if ( event.errorCode ) then
144         native.showAlert( "GPS Location Error", event.errorMessage, {"OK"} )
145         print( "Location error: " .. tostring( event.errorMessage ) )
146     else
147         local locationHandler = function( event )
148             latitudeText = string.format( "%.4f", event.latitude )
149             longitudeText = string.format( "%.4f", event.longitude )
150         end
151         print(latitudeText,longitudeText)
152         local CPFCliente = CPF
153         local path = system.pathForFile("garagens.sqlite", system.DocumentsDirectory)
154         print(path)
155         db = sqlite3.open (path)

```

```

156
157 local tablesetup = "CREATE TABLE IF NOT EXISTS enderecos (ID INTEGER PRIMARY KEY, Endereco, Latitude, Longitude, CPF,pequenos,medios,grandes,motos,pequenos0,medios0,grandes0,motos0,nome);"
158 db:exec(tablesetup)
159 print (tablesetup)
160 local pequeno = vetorCampos[1].text
161 local medio = vetorCampos[2].text
162 local grande = vetorCampos[3].text
163 local moto = vetorCampos[4].text
164 if pequeno == "0" then
165     pequeno = "-1"
166 end
167 if medio == "0" then
168     medio = "-1"
169 end
170 if grande == "0" then
171     grande = "-1"
172 end
173 if moto == "0" then
174     moto = "-1"
175 end
176 local tablefill = "INSERT INTO enderecos VALUES(NULL, '..EnderecoCliente.text..'..'Latitude..'..'Longitude..'..'CPF..'..'pequeno..'..'medio..'..'grande..'..'moto..'..' -1 ..'..' -1 ..'..' -1 ..'..' -1 ..'..' -1 ..'..'Nome da Garagem.text..');"
177 print (tablefill)
178 db:exec(tablefill)
179 db:close()
180 print("db closed")
181
182 end
183 local function cadastralistener( event )
184
185 if ( event.isError ) then
186     local alert = native.showAlert( "Network Error . Check Connection", "Connect to Internet", { "Try again" } )
187     print(event.response)
188
189 else
190 if event.response == "Registrado com sucesso ." then
191     local alert = native.showAlert( "Cadastrando no banco de dados", "Resultado: "..event.response, { "Ok" } )
192
193     EnderecoCliente:removeSelf()
194
195     composer.removeScene( "AdicionarGaragem" )
196     composer.gotoScene("menu", {effect="slideLeft", time = 400})
197 else
198     local alert = native.showAlert( "Erro ao Cadastrar", event.response, { "Try again" } )
199 end
200 end
201
202 end
203 print()
204 local pequeno = vetorCampos[1].text
205 local medio = vetorCampos[2].text
206 local grande = vetorCampos[3].text
207 local moto = vetorCampos[4].text
208 print(pequeno)

```

```

208 if pequeno == "0" then
209     pequeno = "-1"
210 end
211 if medio == "0" then
212     medio = "-1"
213 end
214 if grande == "0" then
215     grande = "-1"
216 end
217 if moto == "0" then
218     moto = "-1"
219 end
220 print(pequeno)
221 print("opa")
222 parameters = {EnderecoCliente.text,latitude,longitude,CPF,pequeno,medio,grande,moto,NomedaGaragem.text}
223 parameters.body = "Register=1&endereco=" .. EnderecoCliente.text .. "&latitude=" .. latitude .. "&longitude=" .. longitude .. "&CPF=" .. CPF ..
224 "&pequenos=" .. pequeno .. "&medios=" .. medio .. "&grandes=" .. grande .. "& motos=" .. moto .. "&pequenos0=" .. "-1" .. "&medios0=" .. "-1" ..
225 "&grandes0=" .. "-1" .. "&motos0=" .. "-1" .. "&nome=" .. NomedaGaragem.text
226 local URL2 = "http://www.coronasdkgames.com/Gabriel/cadastrarEndereco.php"
227 network.request(URL2, "POST", cadastroListener,parameters) -- baixando base de dados completa
228
229
230
231 end
232
233 local Voltar_function = function (event)
234     composer.removeScene( "AdicionarGaragem" )
235     composer.gotoScene("menu", "slideLeft", 400)
236 end
237
238 addButton_Cadastro = widget.newButton{
239     defaultFile = "Button.png",
240     overFile = "Button.png",
241     label = "Cadastrar",
242     fontSize = 44,
243     labelColor = {default = {1,1,1}},
244     emboss=true,
245     onRelease = submitCadastro_function,
246     x = display.contentWidth/2-150,
247     y = display.contentHeight - 200,
248 }
249 addButton_Cadastro.isVisible = false
250 localGroup:insert(addButton_Cadastro)
251
252 local addButton_Voltar = widget.newButton{
253     defaultFile = "Button.png",
254     overFile = "Button.png",
255     label = "Voltar",
256     fontSize = 44,
257     labelColor = {default = {1,1,1}},
258     emboss=true,
259     onRelease = Voltar function,

```

```
260         x = display.contentWidth/2+150,
261         y = display.contentHeight - 200,
262     }
263     localGroup:insert(addButton_Voltar)
264
265 end
266
267
268 function scene:destroy( event )
269     myMap2.isVisible = false
270     local sceneGroup = self.view
271     localGroup:removeSelf()
272
273 end
274
275 scene:addEventListener( "create", scene )
276 scene:addEventListener("destroy", scene)
277
278 return scene
```

Data.lua

```

1 local composer = require("composer")
2 local scene = composer.newScene()
3 local localGroup = display.newGroup()
4 local widget = require("widget")
5
6
7 function scene:create(event)
8     local title = display.newText(localGroup, "Escolha a data", 350, 50, nil, 48)
9     title:setTextColor(1,1,1)
10    local diaLabel = display.newText(localGroup, "dia", 150, 300, nil, 24)
11    diaLabel:setTextColor(1,1,1)
12    local mesLabel = display.newText(localGroup, "mes", 150, 400, nil, 24)
13    mesLabel:setTextColor(1,1,1)
14    local anoLabel = display.newText(localGroup, "ano", 150, 500, nil, 24)
15    anoLabel:setTextColor(1,1,1)
16    local horaLabel = display.newText(localGroup, "hora", 150, 600, nil, 24)
17    horaLabel:setTextColor(1,1,1)
18
19    local dia = native.newTextField(400, 300, 220, 36)
20    localGroup:insert(dia)
21    dia.inputType = "number"
22    local mes = native.newTextField(400, 400, 220, 36)
23    localGroup:insert(mes)
24    mes.inputType = "number"
25    local ano = native.newTextField(400, 500, 220, 36)
26    localGroup:insert(ano)
27    ano.inputType = "number"
28    local hora = native.newTextField(400, 600, 220, 36)
29    localGroup:insert(hora)
30    hora.inputType = "number"
31
32    local submitSelecionar function = function(event)
33        if type(tonumber(dia.text)) ~= "number" or type(tonumber(mes.text)) ~= "number" or type(tonumber(ano.text)) ~= "number" or type(tonumber(hora.text)) ~= "number" then
34            erro = display.newText(localGroup, "preencha todos os campos", 350, 150, nil, 36)
35            erro:setTextColor(1,0,0)
36            localGroup:insert(erro)
37        else
38            if (tonumber(ano.text) % 4 ~= 0) then
39                print("o ano não é bissexto")
40            if (tonumber(mes.text) > 12) then
41                erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
42                erro:setTextColor(1,0,0)
43                localGroup:insert(erro)
44            elseif (tonumber(mes.text) == 1 or tonumber(mes.text) == 3 or tonumber(mes.text) == 5 or tonumber(mes.text) == 7 or tonumber(mes.text) == 9 or tonumber(mes.text) == 10 or tonumber(mes.text) == 12) then
45                if (tonumber(dia.text) > 31) then
46                    erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
47                    erro:setTextColor(1,0,0)
48                    localGroup:insert(erro)
49                else
50                    if (tonumber(ano.text) == dateY) then
51                        if (tonumber(mes.text) < dateM) then
52                            erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)

```

```

53     erro:setTextColor(1,0,0)
54     localGroup:insert(erro)
55 elseif(tonumber(mes.text) == datem) then
56     if(tonumber(dia.text) < dated) then
57         erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
58         erro:setTextColor(1,0,0)
59         localGroup:insert(erro)
60     elseif(tonumber(dia.text) == dated) then
61         if(tonumber(hora.text) <= dateh) then
62             erro = display.newText(localGroup, "A hora selecionada é invalida", 350, 150, nil, 36)
63             erro:setTextColor(1,0,0)
64             localGroup:insert(erro)
65         else
66             datadia = tonumber(dia.text)
67             datames = tonumber(mes.text)
68             dataano = tonumber(ano.text)
69             datahora = tonumber(hora.text)
70             composer.removeScene( "Data" )
71             composer.gotoScene("menu", {effect="slideLeft", time = 400})
72             return true
73         end
74     elseif(tonumber(dia.text) > dated) then
75         datadia = tonumber(dia.text)
76         datames = tonumber(mes.text)
77         dataano = tonumber(ano.text)
78         datahora = tonumber(hora.text)
79         composer.removeScene( "Data" )
80         composer.gotoScene("menu", {effect="slideLeft", time = 400})
81         return true
82     end
83 elseif(tonumber(mes.text) > datem) then
84     datadia = tonumber(dia.text)
85     datames = tonumber(mes.text)
86     dataano = tonumber(ano.text)
87     datahora = tonumber(hora.text)
88     composer.removeScene( "Data" )
89     composer.gotoScene("menu", {effect="slideLeft", time = 400})
90     return true
91 end
92 elseif(tonumber(ano.text) > datey) then
93     datadia = tonumber(dia.text)
94     datames = tonumber(mes.text)
95     dataano = tonumber(ano.text)
96     datahora = tonumber(hora.text)
97     composer.removeScene( "Data" )
98     composer.gotoScene("menu", {effect="slideLeft", time = 400})
99     return true
100 elseif(tonumber(ano.text) < datey) then
101     erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
102     erro:setTextColor(1,0,0)
103     localGroup:insert(erro)
104 end

```

```

105     end
106 elseif(tonumber(mes.text)==2) then
107     if (tonumber(dia.text) > 28) then
108         erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
109         erro:setTextColor(1,0,0)
110         localGroup:insert(erro)
111     else
112         if (tonumber(ano.text) == datey) then
113             if(tonumber(mes.text) < datem) then
114                 erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
115                 erro:setTextColor(1,0,0)
116                 localGroup:insert(erro)
117             elseif(tonumber(mes.text) == datem) then
118                 if(tonumber(dia.text) < dated) then
119                     erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
120                     erro:setTextColor(1,0,0)
121                     localGroup:insert(erro)
122                 elseif(tonumber(dia.text) == dated) then
123                     if(tonumber(hora.text) <= dateh)then
124                         erro = display.newText(localGroup, "A hora selecionada é invalida", 350, 150, nil, 36)
125                         erro:setTextColor(1,0,0)
126                         localGroup:insert(erro)
127                     else
128                         datadia = tonumber(dia.text)
129                         datames = tonumber(mes.text)
130                         dataano = tonumber(ano.text)
131                         datahora = tonumber(hora.text)
132                         composer.removeScene( "Data" )
133                         composer.gotoScene("menu", {effect="slideLeft", time = 400})
134                         return true
135                     end
136                 elseif(tonumber(dia.text) > dated) then
137                     datadia = tonumber(dia.text)
138                     datames = tonumber(mes.text)
139                     dataano = tonumber(ano.text)
140                     datahora = tonumber(hora.text)
141                     composer.removeScene( "Data" )
142                     composer.gotoScene("menu", {effect="slideLeft", time = 400})
143                     return true
144                 end
145             elseif(tonumber(mes.text) > datem) then
146                 datadia = tonumber(dia.text)
147                 datames = tonumber(mes.text)
148                 dataano = tonumber(ano.text)
149                 datahora = tonumber(hora.text)
150                 composer.removeScene( "Data" )
151                 composer.gotoScene("menu", {effect="slideLeft", time = 400})
152                 return true
153             end
154         elseif(tonumber(ano.text) > datey) then
155             datadia = tonumber(dia.text)
156             datames = tonumber(mes.text)

```

```

157         dataano = tonumber(ano.text)
158         datahora = tonumber(hora.text)
159         composer.removeScene( "Data" )
160         composer.gotoScene("menu", {effect="slideLeft", time = 400})
161         return true
162     elseif(tonumber(ano.text) < datey) then
163         erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
164         erro:setTextColor(1,0,0)
165         localGroup:insert(erro)
166     end
167 end
168 elseif(tonumber(mes.text)==4 or tonumber(mes.text)==6 or tonumber(mes.text)==9 or tonumber(mes.text)==11) then
169     if (tonumber(dia.text) > 30) then
170         erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
171         erro:setTextColor(1,0,0)
172         localGroup:insert(erro)
173     else
174         if (tonumber(ano.text) == datey) then
175             if(tonumber(mes.text) < datem) then
176                 erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
177                 erro:setTextColor(1,0,0)
178                 localGroup:insert(erro)
179             elseif(tonumber(mes.text) == datem) then
180                 if(tonumber(dia.text) < dated) then
181                     erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
182                     erro:setTextColor(1,0,0)
183                     localGroup:insert(erro)
184                 elseif(tonumber(dia.text) == dated) then
185                     if(tonumber(hora.text) <= dateh) then
186                         erro = display.newText(localGroup, "A hora selecionada é invalida", 350, 150, nil, 36)
187                         erro:setTextColor(1,0,0)
188                         localGroup:insert(erro)
189                     else
190                         datadia = tonumber(dia.text)
191                         datames = tonumber(mes.text)
192                         dataano = tonumber(ano.text)
193                         datahora = tonumber(hora.text)
194                         composer.removeScene( "Data" )
195                         composer.gotoScene("menu", {effect="slideLeft", time = 400})
196                         return true
197                     end
198                 elseif(tonumber(dia.text) > dated) then
199                     datadia = tonumber(dia.text)
200                     datames = tonumber(mes.text)
201                     dataano = tonumber(ano.text)
202                     datahora = tonumber(hora.text)
203                     composer.removeScene( "Data" )
204                     composer.gotoScene("menu", {effect="slideLeft", time = 400})
205                     return true
206                 end
207             elseif(tonumber(mes.text) > datem) then
208                 datadia = tonumber(dia.text)

```

```

        datames = tonumber(mes.text)
        dataano = tonumber(ano.text)
        datahora = tonumber(hora.text)
        composer.removeScene( "Data" )
        composer.gotoScene("menu", {effect="slideLeft", time = 400})
        return true
    end
elseif(tonumber(ano.text) > datey) then
    datadia = tonumber(dia.text)
    datames = tonumber(mes.text)
    dataano = tonumber(ano.text)
    datahora = tonumber(hora.text)
    composer.removeScene( "Data" )
    composer.gotoScene("menu", {effect="slideLeft", time = 400})
    return true
elseif(tonumber(ano.text) < datey) then
    erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
    erro:setTextColor(1,0,0)
    localGroup:insert(erro)
end
end
end
else
    print("o ano é bissexto")
    if (tonumber(mes.text) > 12) then
        erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
        erro:setTextColor(1,0,0)
        localGroup:insert(erro)
elseif(tonumber(mes.text) == 1 or tonumber(mes.text) == 3 or tonumber(mes.text) == 5 or tonumber(mes.text) == 7 or tonumber(mes.text) == 8 or tonumber(mes.text) == 10 or tonumber(mes.text) == 12) then
    if (tonumber(dia.text) > 31) then
        erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
        erro:setTextColor(1,0,0)
        localGroup:insert(erro)
    else
        if (tonumber(ano.text) == datey) then
            if(tonumber(mes.text) < datem) then
                erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
                erro:setTextColor(1,0,0)
                localGroup:insert(erro)
            elseif(tonumber(mes.text) == datem) then
                if(tonumber(dia.text) < dated) then
                    erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
                    erro:setTextColor(1,0,0)
                    localGroup:insert(erro)
                elseif(tonumber(dia.text) == dated) then
                    if(tonumber(hora.text) <= dateh) then
                        erro = display.newText(localGroup, "A hora selecionada é invalida", 350, 150, nil, 36)
                        erro:setTextColor(1,0,0)
                        localGroup:insert(erro)
                    else
                        datadia = tonumber(dia.text)
                        datames = tonumber(mes.text)

```

```

261     dataano = tonumber(ano.text)
262     datahora = tonumber(hora.text)
263     composer.removeScene( "Data" )
264     composer.gotoScene("menu", {effect="slideLeft", time = 400})
265     return true
266 end
267 elseif(tonumber(dia.text) > dated) then
268     datadia = tonumber(dia.text)
269     datames = tonumber(mes.text)
270     dataano = tonumber(ano.text)
271     datahora = tonumber(hora.text)
272     composer.removeScene( "Data" )
273     composer.gotoScene("menu", {effect="slideLeft", time = 400})
274     return true
275 end
276 elseif(tonumber(mes.text) > datem) then
277     datadia = tonumber(dia.text)
278     datames = tonumber(mes.text)
279     dataano = tonumber(ano.text)
280     datahora = tonumber(hora.text)
281     composer.removeScene( "Data" )
282     composer.gotoScene("menu", {effect="slideLeft", time = 400})
283     return true
284 end
285 elseif(tonumber(ano.text) > datey) then
286     datadia = tonumber(dia.text)
287     datames = tonumber(mes.text)
288     dataano = tonumber(ano.text)
289     datahora = tonumber(hora.text)
290     composer.removeScene( "Data" )
291     composer.gotoScene("menu", {effect="slideLeft", time = 400})
292     return true
293 elseif(tonumber(ano.text) < datey) then
294     erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
295     erro:setTextColor(1,0,0)
296     localGroup:insert(erro)
297 end
298 end
299 elseif(tonumber(mes.text)==2) then
300     if (tonumber(dia.text) > 28) then
301         erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
302         erro:setTextColor(1,0,0)
303         localGroup:insert(erro)
304     else
305         if (tonumber(ano.text) == datey) then
306             if(tonumber(mes.text) < datem) then
307                 erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
308                 erro:setTextColor(1,0,0)
309                 localGroup:insert(erro)
310             elseif(tonumber(mes.text) == datem) then
311                 if(tonumber(dia.text) < dated) then
312                     erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)

```

```

313 erro:setTextColor(1,0,0)
314 localGroup:insert(erro)
315 elseif(tonumber(dia.text) == dated) then
316     if(tonumber(hora.text) <= dateh) then
317         erro = display.newText(localGroup, "A hora selecionada é invalida", 350, 150, nil, 36)
318         erro:setTextColor(1,0,0)
319         localGroup:insert(erro)
320     else
321         datadia = tonumber(dia.text)
322         datames = tonumber(mes.text)
323         dataano = tonumber(ano.text)
324         datahora = tonumber(hora.text)
325         composer.removeScene( "Data" )
326         composer.gotoScene("menu", {effect="slideLeft", time = 400})
327         return true
328     end
329 elseif(tonumber(dia.text) > dated) then
330     datadia = tonumber(dia.text)
331     datames = tonumber(mes.text)
332     dataano = tonumber(ano.text)
333     datahora = tonumber(hora.text)
334     composer.removeScene( "Data" )
335     composer.gotoScene("menu", {effect="slideLeft", time = 400})
336     return true
337 end
338 elseif(tonumber(mes.text) > datem) then
339     datadia = tonumber(dia.text)
340     datames = tonumber(mes.text)
341     dataano = tonumber(ano.text)
342     datahora = tonumber(hora.text)
343     composer.removeScene( "Data" )
344     composer.gotoScene("menu", {effect="slideLeft", time = 400})
345     return true
346 end
347 elseif(tonumber(ano.text) > datey) then
348     datadia = tonumber(dia.text)
349     datames = tonumber(mes.text)
350     dataano = tonumber(ano.text)
351     datahora = tonumber(hora.text)
352     composer.removeScene( "Data" )
353     composer.gotoScene("menu", {effect="slideLeft", time = 400})
354     return true
355 elseif(tonumber(ano.text) < datey) then
356     erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
357     erro:setTextColor(1,0,0)
358     localGroup:insert(erro)
359 end
360 end
361 elseif(tonumber(mes.text)==4 or tonumber(mes.text)==6 or tonumber(mes.text)==9 or tonumber(mes.text)==11) then
362     if (tonumber(dia.text) > 30) then
363         erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
364         erro:setTextColor(1,0,0)

```

```

365     localGroup:insert(erro)
366
367 else
368     if (tonumber(ano.text) == datey) then
369         if(tonumber(mes.text) < datem) then
370             erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
371             erro:setTextColor(1,0,0)
372             localGroup:insert(erro)
373         elseif(tonumber(mes.text) == datem) then
374             if(tonumber(dia.text) < dated) then
375                 erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
376                 erro:setTextColor(1,0,0)
377                 localGroup:insert(erro)
378             elseif(tonumber(dia.text) == dated) then
379                 if(tonumber(hora.text) <= dateh) then
380                     erro = display.newText(localGroup, "A hora selecionada é invalida", 350, 150, nil, 36)
381                     erro:setTextColor(1,0,0)
382                     localGroup:insert(erro)
383                 else
384                     datadia = tonumber(dia.text)
385                     datames = tonumber(mes.text)
386                     dataano = tonumber(ano.text)
387                     datahora = tonumber(hora.text)
388                     composer.removeScene( "Data" )
389                     composer.gotoScene("menu", {effect="slideLeft", time = 400})
390                     return true
391                 end
392             elseif(tonumber(dia.text) > dated) then
393                 datadia = tonumber(dia.text)
394                 datames = tonumber(mes.text)
395                 dataano = tonumber(ano.text)
396                 datahora = tonumber(hora.text)
397                 composer.removeScene( "Data" )
398                 composer.gotoScene("menu", {effect="slideLeft", time = 400})
399                 return true
400             end
401         elseif(tonumber(mes.text) > datem) then
402             datadia = tonumber(dia.text)
403             datames = tonumber(mes.text)
404             dataano = tonumber(ano.text)
405             datahora = tonumber(hora.text)
406             composer.removeScene( "Data" )
407             composer.gotoScene("menu", {effect="slideLeft", time = 400})
408             return true
409         end
410     elseif(tonumber(ano.text) > datey) then
411         datadia = tonumber(dia.text)
412         datames = tonumber(mes.text)
413         dataano = tonumber(ano.text)
414         datahora = tonumber(hora.text)
415         composer.removeScene( "Data" )
416         composer.gotoScene("menu", {effect="slideLeft", time = 400})
417         return true

```

```

417         elseif(tonumber(ano.text) < datey) then
418             erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
419             erro:setTextColor(1,0,0)
420             localGroup:insert(erro)
421         end
422     end
423 end
424 end
425 end
426 end
427
428 local Voltar_function = function (event)
429     composer.removeScene( "Data" )
430     composer.gotoScene("menu", "slideLeft", 400)
431 end
432
433 local addButton_Selecionar = widget.newButton{
434     defaultFile = "Button.png",
435     overFile = "Button.png",
436     label = "Selecionar",
437     fontSize = 44,
438     labelColor = {default = {1,1,1}},
439     emboss=true,
440     onRelease = submitSelecionar_function,
441     x = display.contentWidth/2+150,
442     y = display.contentHeight - 200,
443 }
444 localGroup:insert(addButton_Selecionar)
445
446 local addButton_Voltar = widget.newButton{
447     defaultFile = "Button.png",
448     overFile = "Button.png",
449     label = "Voltar",
450     fontSize = 44,
451     labelColor = {default = {1,1,1}},
452     emboss=true,
453     onRelease = Voltar_function,
454     x = display.contentWidth/2-150,
455     y = display.contentHeight - 200,
456 }
457 localGroup:insert(addButton_Voltar)
458 end
459
460 function scene:destroy( event )
461     local sceneGroup = self.view
462     localGroup:removeSelf()
463 end

```

```

417         elseif(tonumber(ano.text) < datey) then
418             erro = display.newText(localGroup, "A data selecionada é invalida", 350, 150, nil, 36)
419             erro:setTextColor(1,0,0)
420             localGroup:insert(erro)
421         end
422     end
423 end
424 end
425 end
426 end
427
428 local Voltar_function = function (event)
429     composer.removeScene( "Data" )
430     composer.gotoScene("menu", "slideLeft", 400)
431 end
432
433 local addButton_Selecionar = widget.newButton{
434     defaultFile = "Button.png",
435     overFile = "Button.png",
436     label = "Selecionar",
437     fontSize = 44,
438     labelColor = {default = {1,1,1}},
439     emboss=true,
440     onRelease = submitSelecionar_function,
441     x = display.contentWidth/2+150,
442     y = display.contentHeight - 200,
443 }
444 localGroup:insert(addButton_Selecionar)
445
446 local addButton_Voltar = widget.newButton{
447     defaultFile = "Button.png",
448     overFile = "Button.png",
449     label = "Voltar",
450     fontSize = 44,
451     labelColor = {default = {1,1,1}},
452     emboss=true,
453     onRelease = Voltar_function,
454     x = display.contentWidth/2-150,
455     y = display.contentHeight - 200,
456 }
457 localGroup:insert(addButton_Voltar)
458 end
459
460 function scene:destroy( event )
461     local sceneGroup = self.view
462     localGroup:removeSelf()
463 end
464
465 scene:addEventListener( "create", scene )
466 scene:addEventListener("destroy", scene)
467 return scene
468

```

Config.lua

```

1 application =
2 {
3     license =
4     {
5         google =
6         {
7             mapsKey = "AIzaSyDs20N1i2ALps1KuZCG0K33pZt2gxx8M3M",
8         },
9     },
10    content =
11    {
12        width = 720,
13        height = 1280,
14        scale = "letterbox",
15        fps = 30,
16        antialias = false,
17        xalign = "center",
18        yalign = "center"
19    },
20 }

```

Build.Settings

```

1 settings =
2 {
3     orientation =
4     {
5         default = "portrait",
6         content = "portrait",
7         supported =
8         {
9             "portrait",
10        },
11    },
12    android =
13    {
14        usesPermissions =
15        {
16            "android.permission.INTERNET",
17            "android.permission.ACCESS_FINE_LOCATION",
18            "android.permission.ACCESS_COARSE_LOCATION",
19        },
20        usesFeatures =
21        {
22            { name = "android.hardware.location", required = false },
23            { name = "android.hardware.location.gps", required = false },
24            { name = "android.hardware.location.network", required = false }
25        },
26    },
27 },
28 }
29
30

```



```

51 // se inserir obteve sucesso echo usuario e password salvos com sucesso
52 echo "Registrado com sucesso .";
53 }else{
54     echo mysqli_error($con);
55 }
56
57 }
58 }
59 } else{
60     echo "The passwords do not match."; // and send them back to registration page
61 }
62 }
63 }))))
64 -?>

```

CadastrarCartao.php

```

1 <?php
2
3 $con = mysqli_connect("localhost", "coron759_thalesL", "rhodan", "coron759_thalesDB");
4 if(isset($_POST['Register'])) {
5
6     if (empty($_POST["CPF"])) {
7         echo "Fill in CPF to sign up";
8     } else {
9
10        if (empty($_POST["numero"])) {
11            echo "Fill in numero to sign up";
12        } else {
13
14            if (empty($_POST["codigo"])) {
15                echo "Fill in codigo to sign up";
16            } else {
17
18                if (empty($_POST["nome"])) {
19                    echo "Fill in nome to sign up";
20                } else {
21                    if (empty($_POST["validade"])) {
22                        echo "Fill in validade to sign up";
23                    } else {
24
25                        $CPF = mysqli_real_escape_string($con, $_POST["CPF"]);
26                        $numero = mysqli_real_escape_string($con, $_POST["numero"]);
27                        $codigo = mysqli_real_escape_string($con, $_POST["codigo"]);
28                        $nome = mysqli_real_escape_string($con, $_POST["nome"]);
29                        $validade = mysqli_real_escape_string($con, $_POST["validade"]);
30
31                        $result = mysqli_query($con, "SELECT * FROM cartoes WHERE numero='" . $numero . "'");
32
33                        if(mysqli_num_rows($result) > 0)
34                        {
35                            echo "false";
36                        } else {
37
38
39                            $sql = "INSERT INTO cartoes (CPF, numero, codigo,nome,validade) VALUES('" . $CPF . "', '" . $numero . "', '" . $codigo . "', '" . $nome . "', '" . $validade . "')";
40
41                            if(mysqli_query($con, $sql)){
42                                // se inserir obteve sucesso echo usuario e password salvos com sucesso
43                                echo "Registrado com sucesso .";
44                            }else{
45                                echo mysqli_error($con);
46                            }
47                        }
48                    }
49                }
50            }
51        }
52    }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
101 }
102 }
103 }
104 }
105 }
106 }
107 }
108 }
109 }
110 }
111 }
112 }
113 }
114 }
115 }
116 }
117 }
118 }
119 }
120 }
121 }
122 }
123 }
124 }
125 }
126 }
127 }
128 }
129 }
130 }
131 }
132 }
133 }
134 }
135 }
136 }
137 }
138 }
139 }
140 }
141 }
142 }
143 }
144 }
145 }
146 }
147 }
148 }
149 }
150 }
151 }
152 }
153 }
154 }
155 }
156 }
157 }
158 }
159 }
160 }
161 }
162 }
163 }
164 }
165 }
166 }
167 }
168 }
169 }
170 }
171 }
172 }
173 }
174 }
175 }
176 }
177 }
178 }
179 }
180 }
181 }
182 }
183 }
184 }
185 }
186 }
187 }
188 }
189 }
190 }
191 }
192 }
193 }
194 }
195 }
196 }
197 }
198 }
199 }
200 }
201 }
202 }
203 }
204 }
205 }
206 }
207 }
208 }
209 }
210 }
211 }
212 }
213 }
214 }
215 }
216 }
217 }
218 }
219 }
220 }
221 }
222 }
223 }
224 }
225 }
226 }
227 }
228 }
229 }
230 }
231 }
232 }
233 }
234 }
235 }
236 }
237 }
238 }
239 }
240 }
241 }
242 }
243 }
244 }
245 }
246 }
247 }
248 }
249 }
250 }
251 }
252 }
253 }
254 }
255 }
256 }
257 }
258 }
259 }
260 }
261 }
262 }
263 }
264 }
265 }
266 }
267 }
268 }
269 }
270 }
271 }
272 }
273 }
274 }
275 }
276 }
277 }
278 }
279 }
280 }
281 }
282 }
283 }
284 }
285 }
286 }
287 }
288 }
289 }
290 }
291 }
292 }
293 }
294 }
295 }
296 }
297 }
298 }
299 }
300 }
301 }
302 }
303 }
304 }
305 }
306 }
307 }
308 }
309 }
310 }
311 }
312 }
313 }
314 }
315 }
316 }
317 }
318 }
319 }
320 }
321 }
322 }
323 }
324 }
325 }
326 }
327 }
328 }
329 }
330 }
331 }
332 }
333 }
334 }
335 }
336 }
337 }
338 }
339 }
340 }
341 }
342 }
343 }
344 }
345 }
346 }
347 }
348 }
349 }
350 }
351 }
352 }
353 }
354 }
355 }
356 }
357 }
358 }
359 }
360 }
361 }
362 }
363 }
364 }
365 }
366 }
367 }
368 }
369 }
370 }
371 }
372 }
373 }
374 }
375 }
376 }
377 }
378 }
379 }
380 }
381 }
382 }
383 }
384 }
385 }
386 }
387 }
388 }
389 }
390 }
391 }
392 }
393 }
394 }
395 }
396 }
397 }
398 }
399 }
400 }
401 }
402 }
403 }
404 }
405 }
406 }
407 }
408 }
409 }
410 }
411 }
412 }
413 }
414 }
415 }
416 }
417 }
418 }
419 }
420 }
421 }
422 }
423 }
424 }
425 }
426 }
427 }
428 }
429 }
430 }
431 }
432 }
433 }
434 }
435 }
436 }
437 }
438 }
439 }
440 }
441 }
442 }
443 }
444 }
445 }
446 }
447 }
448 }
449 }
450 }
451 }
452 }
453 }
454 }
455 }
456 }
457 }
458 }
459 }
460 }
461 }
462 }
463 }
464 }
465 }
466 }
467 }
468 }
469 }
470 }
471 }
472 }
473 }
474 }
475 }
476 }
477 }
478 }
479 }
480 }
481 }
482 }
483 }
484 }
485 }
486 }
487 }
488 }
489 }
490 }
491 }
492 }
493 }
494 }
495 }
496 }
497 }
498 }
499 }
500 }
501 }
502 }
503 }
504 }
505 }
506 }
507 }
508 }
509 }
510 }
511 }
512 }
513 }
514 }
515 }
516 }
517 }
518 }
519 }
520 }
521 }
522 }
523 }
524 }
525 }
526 }
527 }
528 }
529 }
530 }
531 }
532 }
533 }
534 }
535 }
536 }
537 }
538 }
539 }
540 }
541 }
542 }
543 }
544 }
545 }
546 }
547 }
548 }
549 }
550 }
551 }
552 }
553 }
554 }
555 }
556 }
557 }
558 }
559 }
560 }
561 }
562 }
563 }
564 }
565 }
566 }
567 }
568 }
569 }
570 }
571 }
572 }
573 }
574 }
575 }
576 }
577 }
578 }
579 }
580 }
581 }
582 }
583 }
584 }
585 }
586 }
587 }
588 }
589 }
590 }
591 }
592 }
593 }
594 }
595 }
596 }
597 }
598 }
599 }
600 }
601 }
602 }
603 }
604 }
605 }
606 }
607 }
608 }
609 }
610 }
611 }
612 }
613 }
614 }
615 }
616 }
617 }
618 }
619 }
620 }
621 }
622 }
623 }
624 }
625 }
626 }
627 }
628 }
629 }
630 }
631 }
632 }
633 }
634 }
635 }
636 }
637 }
638 }
639 }
640 }
641 }
642 }
643 }
644 }
645 }
646 }
647 }
648 }
649 }
650 }
651 }
652 }
653 }
654 }
655 }
656 }
657 }
658 }
659 }
660 }
661 }
662 }
663 }
664 }
665 }
666 }
667 }
668 }
669 }
670 }
671 }
672 }
673 }
674 }
675 }
676 }
677 }
678 }
679 }
680 }
681 }
682 }
683 }
684 }
685 }
686 }
687 }
688 }
689 }
690 }
691 }
692 }
693 }
694 }
695 }
696 }
697 }
698 }
699 }
700 }
701 }
702 }
703 }
704 }
705 }
706 }
707 }
708 }
709 }
710 }
711 }
712 }
713 }
714 }
715 }
716 }
717 }
718 }
719 }
720 }
721 }
722 }
723 }
724 }
725 }
726 }
727 }
728 }
729 }
730 }
731 }
732 }
733 }
734 }
735 }
736 }
737 }
738 }
739 }
740 }
741 }
742 }
743 }
744 }
745 }
746 }
747 }
748 }
749 }
750 }
751 }
752 }
753 }
754 }
755 }
756 }
757 }
758 }
759 }
760 }
761 }
762 }
763 }
764 }
765 }
766 }
767 }
768 }
769 }
770 }
771 }
772 }
773 }
774 }
775 }
776 }
777 }
778 }
779 }
780 }
781 }
782 }
783 }
784 }
785 }
786 }
787 }
788 }
789 }
790 }
791 }
792 }
793 }
794 }
795 }
796 }
797 }
798 }
799 }
800 }
801 }
802 }
803 }
804 }
805 }
806 }
807 }
808 }
809 }
810 }
811 }
812 }
813 }
814 }
815 }
816 }
817 }
818 }
819 }
820 }
821 }
822 }
823 }
824 }
825 }
826 }
827 }
828 }
829 }
830 }
831 }
832 }
833 }
834 }
835 }
836 }
837 }
838 }
839 }
840 }
841 }
842 }
843 }
844 }
845 }
846 }
847 }
848 }
849 }
850 }
851 }
852 }
853 }
854 }
855 }
856 }
857 }
858 }
859 }
860 }
861 }
862 }
863 }
864 }
865 }
866 }
867 }
868 }
869 }
870 }
871 }
872 }
873 }
874 }
875 }
876 }
877 }
878 }
879 }
880 }
881 }
882 }
883 }
884 }
885 }
886 }
887 }
888 }
889 }
890 }
891 }
892 }
893 }
894 }
895 }
896 }
897 }
898 }
899 }
900 }
901 }
902 }
903 }
904 }
905 }
906 }
907 }
908 }
909 }
910 }
911 }
912 }
913 }
914 }
915 }
916 }
917 }
918 }
919 }
920 }
921 }
922 }
923 }
924 }
925 }
926 }
927 }
928 }
929 }
930 }
931 }
932 }
933 }
934 }
935 }
936 }
937 }
938 }
939 }
940 }
941 }
942 }
943 }
944 }
945 }
946 }
947 }
948 }
949 }
950 }
951 }
952 }
953 }
954 }
955 }
956 }
957 }
958 }
959 }
960 }
961 }
962 }
963 }
964 }
965 }
966 }
967 }
968 }
969 }
970 }
971 }
972 }
973 }
974 }
975 }
976 }
977 }
978 }
979 }
980 }
981 }
982 }
983 }
984 }
985 }
986 }
987 }
988 }
989 }
990 }
991 }
992 }
993 }
994 }
995 }
996 }
997 }
998 }
999 }
1000 }

```

CadastrarEndereco.php

```

1  <?php
2
3  $con = mysqli_connect("localhost", "coron759_thalesL", "rhodan", "coron759_thalesDB");
4
5  if(isset($_POST['Register'])) {
6
7      if (empty($_POST["endereco"])) {
8          echo"Endereço Incorreto!";
9      } else {
10
11          if (empty($_POST["latitude"])) {
12              echo"Valor da latitude inválido!";
13          } else {
14
15              if (empty($_POST["longitude"])) {
16                  echo"Valor da longitude inválido!";
17              } else {
18
19                  if (empty($_POST["CPF"])) {
20                      echo"CPF inválido!";
21                  } else {
22                      if (empty($_POST["pequenos"])) {
23                          echo"numero pequenos inválido!";
24                      } else {
25                          if (empty($_POST["medios"])) {
26                              echo"numero medios inválido!";
27                          } else {
28                              if (empty($_POST["grandes"])) {
29                                  echo"numero grandes inválido!";
30                              } else {
31                                  if (empty($_POST["motos"])) {
32                                      echo"numero extras inválido!";
33                                  } else {
34                                      if (empty($_POST["pequenos0"])) {
35                                          echo"numero pequenos0 inválido!";
36                                      } else {
37                                          if (empty($_POST["medios0"])) {
38                                              echo"numero medios0 inválido!";
39                                          } else {
40                                              if (empty($_POST["grandes0"])) {
41                                                  echo"numero grandes0 inválido!";
42                                              } else {
43                                                  if (empty($_POST["motos0"])) {
44                                                      echo"numero extras0 inválido!";
45                                                  } else {
46                                                      if (empty($_POST["nome"])) {
47                                                          echo"nome inválido!";
48                                                      } else {
49
50                  $send = mysqli_real_escape_string($con, $_POST["endereco"]);
51                  $latitude = $_POST["latitude"];

```

```

52     $longitude = $_POST["longitude"];
53     $CPF = mysqli_real_escape_string($con, $_POST["CPF"]);
54     $pequenos = mysqli_real_escape_string($con, $_POST["pequenos"]);
55     $medios = mysqli_real_escape_string($con, $_POST["medios"]);
56     $grandes = mysqli_real_escape_string($con, $_POST["grandes"]);
57     $motos = mysqli_real_escape_string($con, $_POST["motos"]);
58     $pequenos0 = mysqli_real_escape_string($con, $_POST["pequenos0"]);
59     $medios0 = mysqli_real_escape_string($con, $_POST["medios0"]);
60     $grandes0 = mysqli_real_escape_string($con, $_POST["grandes0"]);
61     $motos0 = mysqli_real_escape_string($con, $_POST["motos0"]);
62     $nome = mysqli_real_escape_string($con, $_POST["nome"]);
63
64     $sql = "INSERT INTO enderecos (endereco, latitude, longitude,CPF,pequenos,medios,grandes,
65     motos,pequenos0,medios0,grandes0,motos0,nome)
66     VALUES('" . $end . "', '" . $latitude . "', '" . $longitude . "', '" . $CPF . "', '" . $pequenos . "', '" . $medios . "',
67     '" . $grandes . "', '" . $motos . "', '" . $pequenos0 . "', '" . $medios0 . "',
68     '" . $grandes0 . "', '" . $motos0 . "', '" . $nome . "')";
69
70     if(mysqli_query($con, $sql)){
71         // se inserir obteve sucesso echo usuario e password salvos com sucesso
72     echo "Registrado com sucesso .";
73     }else{
74         echo mysqli_error($con);
75     }
76
77 }
78 -}}}}}}}}}}}}}}
79 -?>

```



```

52         }else{
53             if ($tipo == "moto"){
54                 $tipoComp = "motos";
55             }
56         }
57     }
58 }
59 $comparador = mysqli_query($con, "SELECT * FROM reservas WHERE dia='" . $dia. "' AND mes='" . $mes.
60 "' AND ano='" . $ano. "' AND idGaragem='" . $idGaragem. "' AND tipo='" . $tipo. "'");
61 $soma = 0;
62 while ($row = $comparador->fetch_row()) {
63     $soma = $soma + $row[5];
64 }
65
66 // $numMaximo = mysqli_query($con, "SELECT ". $tipo ."s FROM enderecos WHERE id='" . $idGaragem. "'");
67 // $value = mysqli_fetch_object($numMaximo);
68
69 $sql2 = "SELECT ". $tipoComp ." FROM enderecos WHERE id='" . $idGaragem. "'";
70 $result2 = mysqli_query($con, $sql2);
71 $value2 = mysqli_fetch_object($result2);
72 //if ($tipo = "medio"){$tipoComp = "medios";}else{ if ($tipo = "pequeno"){$tipoComp = "medios";}else{if
73 //($tipo = "grande"){$tipoComp = "grandes";}else{if ($tipo = "moto"){$tipoComp = "motos";}}}
74 $numMaximo['myid'] = $value2->$tipoComp;
75
76 if ($numMaximo['myid'] < 0){$numMaximo['myid'] = 0;}
77 if ($soma + $numero > $numMaximo['myid']){
78     if ($numMaximo['myid'] == 0){
79         echo "Esse estabelecimento não possui vagas do tipo " . $tipo . "!";
80     }else{
81         echo "Nesse horário " . $soma . " vagas das " . $numMaximo['myid'] . " do tipo " . $tipo . " já estão reservadas!";
82     }
83 }else{
84     //se inserir obter sucesso echo usuario e password salvos com sucesso
85     $sql = "INSERT INTO reservas (dia, mes, ano, hora, tipo, numero, idGaragem, CPF) VALUES('" . $dia. "', '" . $mes. "',
86     '" . $ano. "', '" . $hora. "', '" . $tipo. "', '" . $numero. "', '" . $idGaragem. "', '" . $CPF. "')";
87     if(mysqli_query($con, $sql)){
88         echo "Registrado com sucesso.";
89     }else{
90         echo mysqli_error($con);
91     }
92 }
93 }
94 }
95 }
96 )))))))
97 ?>

```

ImportarBD.php

```

1 <?php
2 //
3 // getstatus.php for the Trivia Game
4 //
5
6
7 $con = mysqli_connect("localhost", "coron759_thalesL", "rhodan", "coron759_thalesDB");
8
9
10 mysqli_query($con, "SET CHARACTER SET utf8");
11 mysqli_query($con, "SET NAMES utf8");
12
13 $tipoEntrada = mysqli_real_escape_string($con, $_POST["tipo"]);
14
15 if($tipoEntrada == "cadastros") {
16     $result = mysqli_query($con, "SELECT * FROM cadastros");
17 }else{
18 if($tipoEntrada == "cartoes"){
19     $result = mysqli_query($con, "SELECT * FROM cartoes");
20 }else{
21 if($tipoEntrada == "enderecos"){
22     $result = mysqli_query($con, "SELECT * FROM enderecos");
23 }
24 }
25 }
26
27 if ($result) {
28     if(mysqli_num_rows($result)){
29         while($row=mysqli_fetch_row($result)){
30             $json[]=$row;
31         }
32     }
33     echo json_encode($json,SON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE);
34 }else{
35     echo "null";
36 }
37
38
39 ?>

```

BuscarReserva.php

```

1 <?php
2 ///////////////////////////////////////////////////
3 // Connecting, selecting database
4 ///////////////////////////////////////////////////
5 // $link = mysql_connect("localhost", "coron759_thalesL", "rhodan") or die('Could not connect: ' . mysql_error());
6 $con = mysqli_connect("localhost", "coron759_thalesL", "rhodan", "coron759_thalesDB");
7
8 ///////////////////////////////////////////////////
9 if(isset($_POST)) {
10     $CPF= mysqli_real_escape_string ($con,$_POST["CPF"]);
11
12     mysqli_query($con, "SET CHARACTER SET utf8");
13     mysqli_query($con, "SET NAMES utf8");
14
15     $result = mysqli_query($con, "SELECT * FROM reservas WHERE CPF=" . $CPF); //. " AND mes=" . $mes . " AND ano = " . $ano . " AND hora = " . $hora);
16     $rowcount = mysqli_num_rows($result);
17
18     if ($result){
19         if(mysqli_num_rows($result)){
20             while($row=mysqli_fetch_row($result)){
21                 $json[]=$row;
22             }
23             $json["quantidade"] = $rowcount
24         }
25         echo json_encode($json,SON_UNESCAPED_SLASHES | JSON_UNESCAPED_UNICODE);
26     }else{
27         echo "null";
28     }
29 }
30 ?>

```